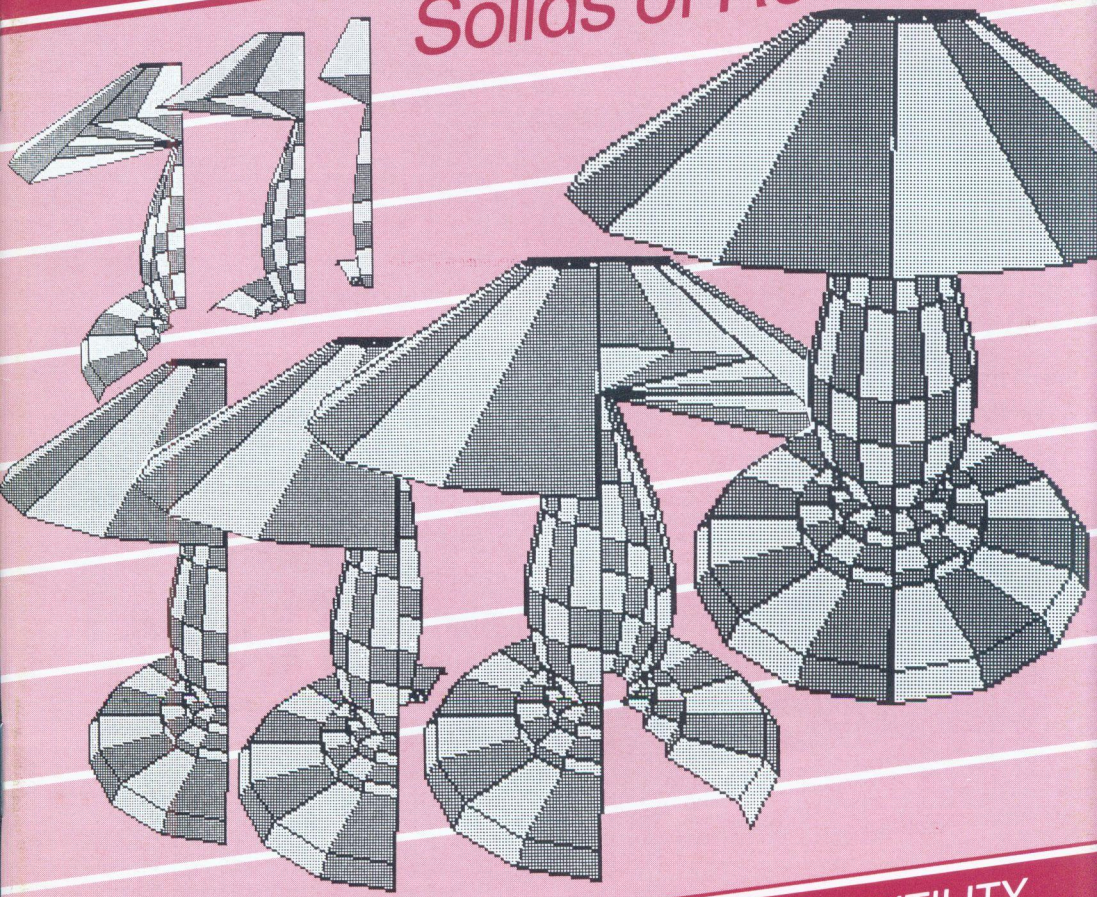
Vol.8 No.7 December 1989

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

Solids of Revolution



- LAUNCH A SPACECRAFT
- A FIND UTILITY
- POSTSCRIPT VECTOR GRAPHICS
- EDIKIT

BEEBUG Vol.8 No.7 December 1989

FEATURES

REVIEWS

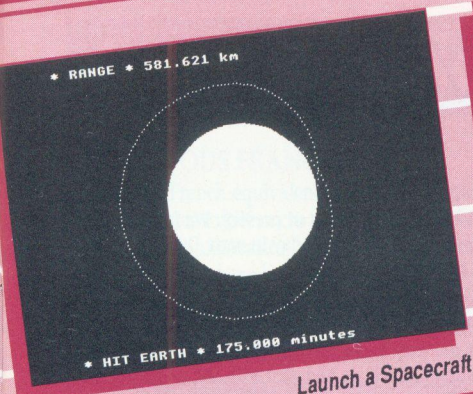
REGULAR ITEMS

HINTS & TIPS

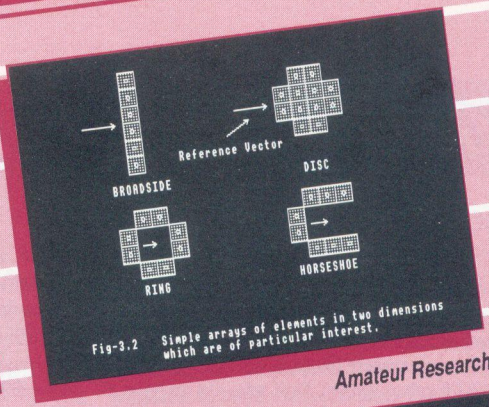
PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

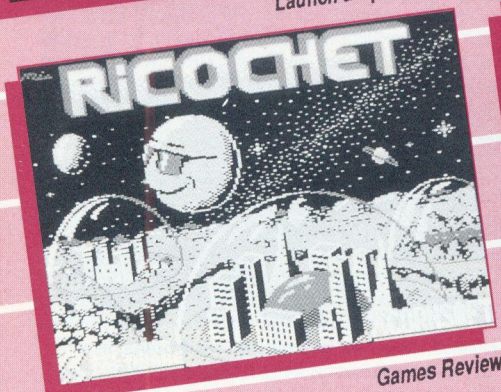
All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints



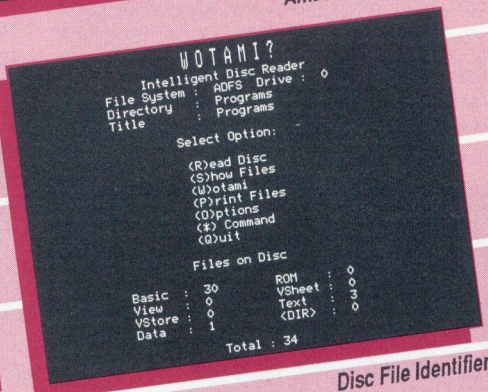
Launch a Spacecraft



Amateur Research



Games Review



Disc File Identifier

Solids of Revolution

Postscript Vector Graphics

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BEEBUG SUMMER COMPETITION

We were very pleased to welcome the winner of our Summer Competition, Paul Warren, and his wife to BEEBUG in November where Paul was able to collect his prize of an A3000 colour system. Paul, who has an issue 3 model B, was highly delighted at his good fortune, and was looking forward to an increased interest in home computing with his new system. Paul claims no great mathematical ability, but says once he started on the competition, he just became 'hooked'. Our congratulations to Paul.

COMPUTER SHOPPER SHOW

As a result of a last minute decision, BEEBUG did in the end have a stand at the Computer Shopper Show on 24th-26th November 1989. The show, at least on the Sunday when I attended, was absolutely packed throughout the day, and indeed by 4.30pm, half an hour after the official closing time, the show was still very busy. There were stands from a number of other companies active in the Acorn market, Minerva Software for example, but their presence was scattered among the many stands more devoted to Amstrad, Atari and Commodore products. Given more constructive support from Acorn and a better concentration of Acorn related stands, future Computer Shopper Shows could be well worth attending, if you can survive the sheer crush of bodies that is.

A more relaxed way of seeing the best and latest in Acorn and related products is to attend the open days organised by BEEBUG and others. This seems to be a growing feature of the Acorn market at present. We hope that those who attended our own open day in early December found the day rewarding.

BEEBUG READER SURVEY

Many of you may be wondering what happened to the reader survey which we included with the magazine earlier this year. We received well over 500 completed forms which is an excellent response and indicates the high level of commitment of many members to BEEBUG. The survey has provided us with a wealth of information which we are still analysing. Some of the additional comments of readers are also included this month on our Postbag page.

As stated there, BEEBUG will continue to support users of the model B, Master and Compact for as long as demand exists, and indeed as some other magazines jump increasingly on the Archimedes bandwagon, we may see BEEBUG assuming even greater importance in the lives of BBC micro owners. However exciting the Archimedes range is, it is a sobering thought that sales have still to reach the estimated 1.5 million BBC micros sold. One wonders where many of these machines have gone, though it is interesting how often one gets a glimpse of a BBC micro on TV still doing sterling work. There is obviously a lot of life left yet in what is still an excellent machine.

STAFF CHANGES

Astute readers may have noticed from the list of credits that Technical Editor David Spencer is no longer with BEEBUG magazine. However, he has not left the company, and is now in charge of hardware development for BEEBUG, so we are likely to see increased activity in this area.

His position as Technical Editor has now been taken over by Alan Wrigley who joined BEEBUG at the start of December. Alan has extensive experience with both the BBC and Archimedes ranges, and has written for other magazines on a freelance basis in the past.

Finally, assuming that this issue reaches you before Christmas, as is our intention, may I wish you the season's greetings from all the magazine staff, and look forward with you to the new year. Remember that the next issue of BEEBUG is a two month issue for January and February, and is currently scheduled for publication at the end of January.

News News News News News News

PARLEZ VOUS FRANCAIS?

Minitel, the French equivalent of Britain's Prestel service, is now available to UK micro users at low cost. Minitel is a computerised information service currently accessed by terminal and telephone line from over 5 million homes and offices in France. Unlike the UK, the French government has subsidised the service by giving terminals away free, and the service has rocketed to success.

Now, Aldoda International can provide access to the entire range of Minitel services at local call cost in the UK. The Minitel software supplied by Aldoda costs from £28.50, with versions available for BBC/Acorn machines, and for PCs and compatibles. As an added bonus, many Minitel services are available in both French and English.

For more information contact Aldoda International Ltd., 27 Elizabeth Mews, London NW3 4UH, tel. 01-586 5686.

CHEQUED BY COMPUTER

New software for the BBC micro was released earlier this year by newcomer Carious Software in the form of *Cheque*, a software package intended to simplify the generation, printing and analysing of cheques for a club or small business. It is intended for situations where payments are made regularly or frequently to a small list of suppliers whose details can be kept in a separate file. The package not only prints the cheque, but also prints a self adhesive label for the envelope. The program will also list, analyse and summarise cheques written over a period of time.

Cheque costs £24.95 inc VAT and p&p, and is available from Carious Software, 54A Moor Crescent, Gosforth, Newcastle upon Tyne NE3 4AQ, tel. (091) 285 0097.

NEW RELEASES AND SPECIAL OFFERS FROM TOPOLOGIKA

Early in 1990, Tolpologika will release the second in its popular *Catch Up With The national Curriculum* software series. Hot on the heels of *Punctuate* comes *Untrivia - Catch Up With Your Maths*. This is a quiz for one or two children in multi-choice format. The System Disc costs £9.95 ex. VAT, and this includes a sample general knowledge quiz, while the maths quiz discs are available separately (Key stages 2, 3 and 4), also at £9.95 each.

A second release for the science curriculum is *Astro - The Earth in Space*, described as an interactive encyclopaedia from which to learn about the movements of the sun, moon, planets, comets, spacecraft etc. at a yet to be announced price. This follows *Whale Facts* already released for the BBC micro and costing £16.00 ex. VAT.

Topologika also has some special offers on its games range (Avon, Acheton, Kingdom of Hamil, Countdown to Doom, Return to Doom etc.) all at £11.95 inc. VAT until 28th February, with further discounts if you order two or more at the same time.

Finally, Topologika has announced a link-up with Australian software house Amig Systems. Amig will distribute Topologika software products down under, while Topologika market selected Amig products 'up top'. These include *Account Ability* to teach small business accounts, and supplied in two packs (Documents Module and Journal Module) at £29.95 ex. VAT each. For more information, contact Topologika at P.O.Box 39, Stilton, Peterborough PE7 3RL, tel. (0733) 244682.

TOMORROW IS HERE NOW

The Rutherford Appleton Laboratory in Oxfordshire has released a disc of programs and accompanying leaflet entitled 'In Search of Tomorrow' to publicise its work in computing, and to support the new Information Technology in schools. Depending on feedback, other discs could follow. Schools who would like a copy of the disc, which contains 15 programs and supporting files (including three programs supplied by BEEBUG magazine), should contact the Press and Publicity Section, Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, tel. (0235) 21900.

The New Master Mega-ROM

David Spencer looks at a revised operating system for the Master 128.

Product	Alternative Master MOS
Supplier	Acorn Computers Ltd., Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN. Tel. (0223) 245200
Price	£44.85 (inc. VAT) £42.71 to BEEBUG members

Just as many were saying that Acorn only cared about the Archimedes and beyond, the company has bitten back by releasing a revised operating system for the Master 128. The *alternative ROM*, as Acorn refers to the new MOS, is a plug in replacement for the standard ROM, and updates all the system software with the exception of the ANSI terminal emulator *Terminal*.

INSTALLATION

Installing the alternative ROM is little different to installing any other ROM inside the Master. Having removed the top cover of the computer, you have to identify and remove the system ROM from its socket. On some early systems, this is attached to a small carrier board, and in a very few it is soldered directly to the board. In this latter case, the machine must be taken to an Acorn dealer who will unsolder the ROM and install a socket in its place. However it's done, once the old ROM has been removed, the new one simply plugs in, and that is the installation operation over.

CHANGES AND NEW FEATURES

There are far too many changes and new features incorporated in the alternative ROM to list them all here. Most are either minor, such as modifying *SHOW to display all the function keys if no argument is given, or fix bugs in the original MOS, such as the infamous CLOSE #0 bug in the DFS (see last month's BEEBUG). However, there are a number of major changes and additions.

Perhaps the area of most change is ADFS. For one thing, this has speeded up by a factor of at least two when using floppies. Furthermore, the Format, Verify and Backup utilities are now included in the ROM, meaning that you can throw away the dreaded Master utility disc. *COMPACT within ADFS has also been simplified. You no longer need to specify an area of memory that can be used, and the call does not corrupt the user workspace. DFS has been modified to speed up the OSGBPB call and to allow files longer than 64K to be saved.

Basic has been modified by using improved floating point arithmetic and trigonometry routines. These can result in appreciable speed increases in programs that perform a lot of floating point computation.

View now features a built-in Epson printer driver (as on the Master Compact) and also allows the justification, formatting and insert/overtype status to be configured in CMOS RAM. Edit has been extended to allow files to use either carriage returns or line feeds as terminators, and a new key command has been added to switch between these. The annoying feature that Return had to be pressed after entering the mode number for the change mode command has been removed, and a print 'As Is' option has been added to bypass the normal formatting commands that are applied when text is printed. Finally for Edit, an 'End of file replace' option has been added to stop search and replace from prompting for each change. This is particularly useful when specifying a complex matching template, as you can try it on the first few occurrences, and then apply it globally to the rest of the file when you have checked that it is correct.

Another major change is the addition of support for 8-bit keyboard codes. Basically, this allows international characters to be entered from the keyboard. However, the alternative ROM does not, unlike the Master Compact, provide this directly. (The Compact has a Code key which allows international characters to be entered.) Instead, it offers a set of 'hooks' which attach to another ROM, called the

International ROM. Acorn will make this ROM available if it is needed.

Finally, all the known bugs in the standard MOS ROM have been fixed.

THE RELOCATOR

The Relocator is one feature new to the alternative ROM which will only be of benefit to owners of 6502 second processors (or the Master Turbo board). To understand its purpose, and how it works, you need to look at the way in which the memory map of the Master differs from that of the second processor.

In the Master, any 16K language ROM occupies the memory between addresses &8000 and &BFFF, with the memory from &C000 to &FFFF being used for the operating system. On a second processor, the operating system is only 2K long, and occupies memory from &F800 to &FFFF. Hence, if a ROM is copied to the second processor at its address of &8000, the 14K between &C000 and &B7FF is wasted. What is needed is to move the ROM image up to start at address &B800. This will close the gap, and allow the extra memory below the image to be used. However, the situation is not as simple as this, as instructions in the ROM image that refer to absolute addresses within that image must be changed. This is the function of the Relocator, which uses a 'bit-map' to determine for each byte in the ROM image whether it needs to be altered or not, and by how much. All this is done automatically as the image is transferred to the second processor.

The alternative ROM uses the Relocator to move-up Basic and Edit (eliminating the need for HiBasic and HiEdit), but not Viewsheet. View includes its own relocator which performs a similar function.

COMPATIBILITY

Acorn have decided not to include the alternative ROM as standard in new Masters. The reason, they say, is that while the new ROM is highly compatible, there will be some existing programs that don't work with it. In particular, the new 8-bit key handling may confuse some programs, and any program which directly accesses routines within the ROM will almost certainly fail. I feel that Acorn have made a wise decision, considering the amount of software already available for the Master, but I don't think that you should be put off by incompatibility - in practice most programs will work.

DOCUMENTATION

The alternative ROM is accompanied by a 26-page A5 user guide, split into four sections. The first of these is an introduction including fitting instructions, while the second and third sections describe the changes and new features from the user's point of view, and a technical view point respectively. The final section explains the workings of the Relocator, and shows how you can write self-relocating ROM images. The listing of a program to do this for you automatically is also included.

The documentation is designed to be used in conjunction with the two-part *Master Series Reference Manual*, and is totally adequate for describing the alternative ROM.

CONCLUSION

Acorn should have released the alternative ROM a year ago - a move that would not have been impossible, as a number of people, myself included, had test versions as long as two and a half years ago. It is a fact of progress that a large number of the more 'active' Master users, who would have probably been the first to buy this product, have now shelved their machines and moved onto pastures new, such as the Archimedes. However, having said that, I feel that Acorn have done themselves no harm releasing the new ROM now, and they may well have caught out a few of their critics.

Whether you actually need the alternative ROM or not depends a lot on what you use your Master for. In some cases you may not benefit at all, and may even suffer if you spend most of the time using one of the very few packages that is incompatible with the alternative ROM. On the other hand, I would say that both second processor owners and regular ADFS users should put their order in the post today as they will benefit highly from the new features.

Launch a Spacecraft

by Donald Tattersfield

The reception of television programmes via satellites which are in orbit round the Earth is a reality today. Other satellites, of which you will have heard, are those which help in weather forecasting, and those used in civil and military surveying. There are hundreds more.

The program listed here allows you to simulate the launching of such a spacecraft into an orbit of your own choosing. The method used is very simple indeed, but you do not need to understand it in order to enjoy the program. The program can be as addictive as an arcade game, so beware!

GENERAL INFORMATION

We need to find the cube of the distance of the spacecraft from the centre of the Earth, so to keep the numbers relatively small we have used the radius of the Earth as our unit of distance, instead of the more familiar mile or km. Thus 1 e.r. is the same as 6378 km. In keeping with this we have used e.r./min as the unit of velocity, which is the same as 106.3 km/s. This information, and what some of the variables mean are shown in the REM statements in lines 140-280, and in figure 1.

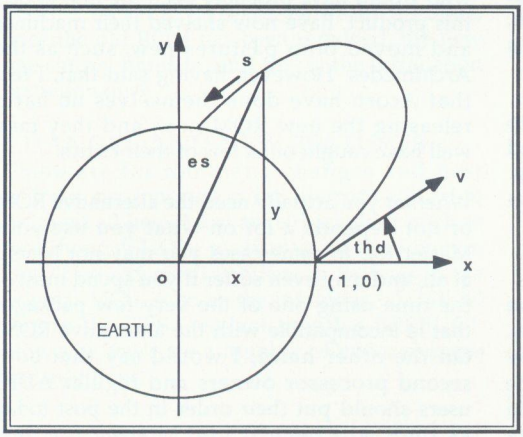


Figure 1

USING THE PROGRAM

Type in the program, save it and then run it. For a first try you might send up your spacecraft in such a way that it comes down to Earth again. You may remember that in developing the equipment for sending a spacecraft to the Moon, the Gemini and Mercury spacecraft were first tried out with such an orbit.

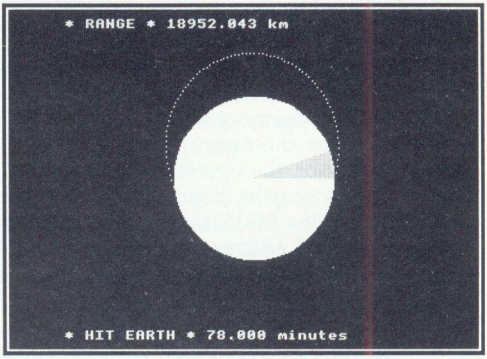


Figure 2

You will be prompted to input the position from which the launch takes place. The X and Y co-ordinates are the distances from the centre of the Earth, measured in the Ox and Oy directions respectively (see figure 1). Enter (1,0) so that you will be launching from the surface of the Earth (refer to figure 1 again). For velocity enter 0.08 (which is equivalent to 8.5 km/min). Use a launch angle of 70 degrees and a scale of 250 for your graphics. At the moment we are going to allow the spacecraft to move under the gravitational attraction of the Earth only for the whole of the flight, so reply 'N' when asked if you wish to make any corrections. Finally, enter 1 minute for the chosen time interval. The resulting path of the spacecraft is shown in figure 2. Notice that the Earth is still revolving while the spacecraft is in flight, and this is taken into account when calculating the range.

For further orbits, alter one of the input values, keeping the rest the same, until you are familiar with the program. You might be surprised that by altering the time interval only, you appear to get different orbits. The explanation of why this is so is given in the Technical Notes below.

Some spacecraft carry the means to alter their velocity while in flight, and you can do so with your spacecraft if you reply 'Y' when asked to opt for corrections. It is easiest to see the effect of altering vx or vy when the spacecraft is moving parallel to one of the axes. Watch the spacecraft in the example given above, and note the time when this occurs. On a re-run, input a correction at this time, say, 39 min, vx=-0.06, vy=0 and you will see the spacecraft change its orbit into an oval (nearly a circle) round the Earth (see figure 3). You can abort the mission at any time by pressing key 'Q'. Any correction time must be an exact multiple of the chosen time interval. See if you can add a second correction to bring the spacecraft down near to the point of launch (see figure 3).

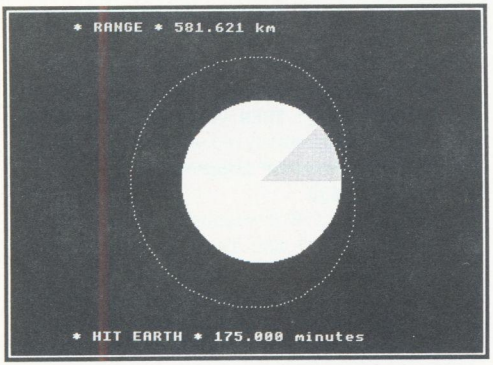


Figure 3

At any time, pressing Escape will abort the current orbit and return you to the data entry screen, while Shift-Escape will exit from the program completely.

GEO-SYNCHRONOUS ORBIT

This type of orbit occurs when a spacecraft revolves round the Earth once in a day. This means that the spacecraft appears to sit above the same point on the Earth's surface, which is the key to television satellites. To see such an orbit, input (xo,yo) as (6.6,0), v as 0.02895, angle as 90 degrees, scale=60 and dt=2 minutes with no corrections.

TECHNICAL NOTES

To simplify the program it is assumed that the Earth is a uniform sphere, and that launch takes place in the plane of the equator. The display of the orbit is as seen from a point above the North Pole. Newton's law of gravitation has been applied to each of the component axes Ox, Oy. For the rest, all we have assumed is that for short time intervals the component accelerations ax, ay do not change.

We may then apply two of the laws of motion with constant acceleration to obtain successive values of x, y the co-ordinates of the spacecraft and the corresponding vx, vy. It will be clear that the smaller we make dt the more correct will be the orbit. We would suggest halving the time interval chosen until no appreciable difference is seen in successive orbits. Finally, as with many numerical methods, starting the system requires special conditions, and these are given in PROCstart.

CONCLUSION

Now that you have seen the possibilities of the program, you can set your own targets - there are literally hundreds of orbits you can try!

```
  10 REM Program SpaceB
  20 REM Launch a spacecraft
  30 REM Version B1.2
  40 REM Author  Donald Tattersfield
  50 REM BEEBUG  December 1989
  60 REM Program subject to copyright
  70 :
 100 MODE 7:ON ERROR GOTO880
 110 PROCtitle
 120 DIM t(4),vx(4),vy(4),pt(60,2)
 130 cc=PI/180
 140 REM Units used
 150 REM me=Mass of Earth=1 e.m.(5.97E2
4 kg)
 160 me=1
 170 REM re=Radius of Earth=1 e.r.(6378
```

```
 km)
  180 re=1
  190 REM Time unit=1 minute
  200 REM Velocity unit=1 e.r./minute (1
06.3 km/s)
  210 REM G=Gravitational constant
  220 G=0.00553
  230 REM es=Distance of spacecraft from
 centre of Earth
  240 REM scale=Scale of display
  250 REM rate=Rate of rotation of Earth
 (rad/min)
  260 rate=2*PI/1440
  270 REM Spacecraft image
  280 VDU 23,224,128,0,0,0,0,0,0
  290 :
  300 MODE 7
  310 VDU131:PRINT"** INPUT LAUNCH CONDI
TIONS **"
  320 PRINT:PRINT
  330 VDU 130:INPUT "LAUNCH X CO-ORDINAT
E (e.r.)",xo
  340 VDU 130:INPUT "LAUNCH Y CO-ORDINAT
E (e.r.)",yo
  350 PROClaunchangle
  360 PRINT
  370 VDU 131:INPUT "LAUNCH VELOCITY (e.
r./min)",v
  380 PRINT
  390 VDU 133:INPUT "LAUNCH ANGLE (degre
es to x-axis)",thd
  400 thr=thd*cc
  410 PRINT
  420 VDU 130:INPUT "Scale",scale
  430 PRINT
  440 VDU 135:PRINT"Do you want to make"
  450 VDU 135:INPUT "flight corrections
(Y/N)",flight$
  460 PRINT
  470 IF flight$="Y" OR flight$="y" THEN
 PROCcorrections
  480 VDU 134:INPUT "Chosen time interva
l (minutes)",dt
  490 CLS
  500 :
  510 REM Special conditions for start
  520 PROCstart
  530 MODE 1
  540 VDU 19,3,4,0,0,0
  550 REM Draw Earth
  560 cx=600:cy=500:r=scale
  570 xoldearth=cx+re*COS(launchangle):y
oldearth=cy+re*SIN(launchangle)
  580 GCOL 0,3:PROCcircle(cx,cy,r)
  590 REM Normal conditions
  600 VDU 5
  610 REM **Start of loop**
  620 REM Print time elapsed (min)
  630 PROCprinttime
  640 t=t+dt
  650 IF flight$<>"Y" THEN 690
  660 FOR I=1 TO nc
  670 IF t=t(I) THEN vx=vx(I):vy=vy(I)
  680 NEXT I
  690 es=SQR(x*x+y*y):es3=es^3
  700 sax=ax:say=ay
  710 IF es<re THEN VDU 7
  720 IF es<re THEN VDU 5:GCOL 0,2:PRINT
"* HIT EARTH *";" ";t;" ";"minutes"
  730 IF es<re THEN PRINT:PROCrange:PROC
repeat:GOTO 300
  740 ax=-G*me*x/es3:ay=-G*me*y/es3
  750 dax=(ax-sax)/dt:day=(ay-say)/dt
  760 x=x+vx*dt+ax*dt*dt/2
  770 y=y+vy*dt+ay*dt*dt/2
  780 vx=vx+ax*dt+dax*dt*dt/2
  790 vy=vy+ay*dt+day*dt*dt/2
  800 REM Print spacecraft position
  810 PROCrotateearth
  820 PROCspacecraft
  830 Q$=INKEY$(0)
  840 IF Q$="Q" THEN VDU 4:PROCrepeat:GO
TO 300
  850 REM **End of loop**
  860 GOTO 630
  870 :
  880 IF ERR=17 AND INKEY-1 MODE7:END
  890 IF ERR=17 THEN GOTO300
  900 MODE7:REPORT:PRINT" at line ";ERL
  910 END
  920 :
 1000 DEF PROCstart
 1010 t=0
 1020 vxo=v*COS(thr):vyo=v*SIN(thr)
 1030 ax=0:ay=0:sax=0:say=0
 1040 x=xo:y=yo:vx=vxo:vy=vyo
 1050 ENDPROC
 1060 :
 1070 DEF PROCcircle(cx,cy,r)
 1080 diffangle=2*PI/60
 1090 cq=COS(diffangle):sq=SIN(diffangle
)
```

```
1100 pt(1,1)=cx+r:pt(1,2)=cy
1110 PLOT 69,pt(1,1),pt(1,2)
1120 FOR I=2 TO 60
1130 xd=pt(I-1,1)-cx:yd=pt(I-1,2)-cy
1140 pt(I,1)=cx+xd*cq-yd*sq
1150 pt(I,2)=cy+xd*sq+yd*cq
1160 MOVE cx,cy
1170 PLOT 85,pt(I,1),pt(I,2)
1180 NEXT I
1190 pt(60,1)=pt(1,1):pt(60,2)=pt(1,2)
1200 MOVE cx,cy
1210 PLOT 85,pt(60,1),pt(60,2)
1220 GCOL 0,2:PLOT 69,cx,cy
1230 ENDPROC
1240  :
1250 DEF PROCspacecraft
1260 MOVE 600+scale*x,500+scale*y
1270 GCOL 0,2:PRINT CHR$(224)
1280 ENDPROC
1290 :
1300 DEF PROCprinttime
1310 @%=&20309:GCOL 0,2
1320 MOVE 950,60:PRINT t
1330 GCOL 0,0
1340 MOVE 950,60:PRINT t
1350 ENDPROC
1360 :
1370 DEF PROCrotateearth
1380 IF t>1440 AND t<=2880 THEN GCOL 0,
1 ELSE GCOL 0,2
1390 MOVE cx,cy
1400 DRAWxoldearth,yoldearth
1410 xearth=cx+scale*COS(launchangle+t*
rate)
1420 yearth=cy+scale*SIN(launchangle+t*
rate)
1430 xoldearth=xearth:yoldearth=yearth
1440 PLOT 85,xearth,yearth
1450 ENDPROC
1460 :
1470 DEF PROCrange
1480 IF x=0 THEN x=0.0001
1490 impactangle=ATN(y/x)
1500 IF (y>0 AND x<0) OR (y<0 AND x<0)
THEN impactangle=impactangle+PI
1510 IF y<0 AND x>0 THEN impactangle=im
pactangle+2*PI
1520 PRINT"* RANGE *";" ";6378*(impacta
ngle-launchangle-t*rate);" ";"km"
1530 ENDPROC
1540 :
```

```
1550 DEF PROCcorrections
1560 PRINT
1570 VDU 135:PRINT"How many corrections
"
1580 VDU 135:INPUT"do you wish to make
(max.4)",nc
1590 PRINT
1600 VDU 131:PRINT"INPUT time elapsed (
min from launch)"
1610 VDU 131:PRINT"INPUT new vx (e.r./m
in)"
1620 VDU 131:PRINT"INPUT new vy (e.r./m
in)"
1630 PRINT
1640 FOR I=1 TO nc
1650 VDU 131:INPUT t(I)
1660 VDU 131:INPUT vx(I)
1670 VDU 131:INPUT vy(I)
1680 PRINT
1690 NEXT I
1700 PRINT
1710 ENDPROC
1720 :
1730 DEF PROCtitle
1740 t$=STRING$(17,CHR$(94)+" ")
1750 PRINTTAB(2,8) CHR$131 t$
1760 PRINTTAB(2,18) CHR$131 t$
1770 PRINTTAB(10,10);"LAUNCH A SPACECRA
FT"
1780 PRINTTAB(7,15);CHR$134;"by Donald
Tattersfield"
1790 TIME=0:REPEAT UNTIL TIME>200:CLS
1800 ENDPROC
1810 :
1820 DEF PROClaunchangle
1830 IF xo=0 THEN xo=0.0001
1840 launchangle=ATN(yo/xo)
1850 IF xo<0 AND yo=0 THEN launchangle=
PI
1860 IF (xo<0 AND yo>0) OR (xo<0 AND yo
<0) THEN launchangle=launchangle+PI
1870 IF xo>0 AND yo<0 THEN launchangle=
launchangle+2*PI
1880 ENDPROC
1890 :
1900 DEF PROCrepeat
1910 PRINT:GCOL 0,1
1920 INPUT "Another orbit (Y/N)",orbit$
1930 IF orbit$="Y" OR orbit$="y" THEN 1
940 ELSE END
1940 ENDPROC
```

Edikit (Part 1)

This month, Bill Hine begins a new series which will build up an indispensable utility ROM for all Basic programmers.

As your Basic program gets longer and longer, so finding your way round it gets more and more difficult. Have you ever had problems remembering the names of the variables you have used, or of finding the right program line to edit? Have you ever wanted to look at the code for a given function or procedure, or to list out a FOR loop but not been able to locate it? And what about changing a variable name throughout a program, or finding out how much memory is still left for your program?

The answer to these and many more problems is the EDIKIT ROM for the Master, BBC B or B+ (all with sideways RAM). Containing eleven commands to help with the editing and development of Basic programs, EDIKIT is also designed to be expandable. It contains features which make it easy to add your own commands to the ROM or to incorporate other published programs. For example, a future article will explain how David Spencer's excellent Partial Renumber commands (BEEBUG Vol.7 No.7) may be added to EDIKIT.

The ROM includes three FIND commands designed to help locate lines of program according to their content; three LIST commands list out significant segments of a program; two REPLACE commands enable you to make alterations throughout your program. Finally there are three miscellaneous commands giving background information on system variables and sizes (values of PAGE, HIMEM etc., and the sizes of your program and the remaining free memory), a list of variable names and a print out of current function key definitions.

ENTERING THE PROGRAM

The listing given here is EDIKIT1, the first of four programs making up the core of EDIKIT. It contains the ROM header and the code for the *FTEXT (FIND TEXT) command together with many of the routines which will form the basis of the remaining commands to be presented in further articles in this series.

There are no special problems in entering the program except to note that the value of ttadr% is &8071 for Basic II and &8456 for Basic IV (lines 130 and 140). If your system does not recognise the standard Acorn command *SRLOAD, for loading a ROM image into sideways RAM, you will also need to substitute the appropriate command for your sideways RAM in line 200.

USING EDIKIT1

After entering the program (necessarily rather long as it provides the framework for all subsequent functions as well), save it as EDIKIT1. Now run it, respond Y to the prompt "SAVE and LOAD ROM?" (the image is automatically loaded into slot 5), and then press Ctrl-Break (or enter *INIT, if you are fortunate enough to own the BEEBUG Master ROM). A copy of the ROM image will be saved as EDIROM1, and may be subsequently reloaded with:

```
*SRLOAD edirom1 8000 <rom-id>
```

(or your corresponding command). You should find that typing *HELP EDIKIT will print a list of all the commands recognised by the new ROM image (at present just one *FTEXT) and their parameters. Enter OLD and then:

```
*FTEXT /ptr/
```

or:

```
*ft./ptr/
```

Most EDIKIT commands may be abbreviated to one or two characters followed by a full stop; you can also use any combination of upper and lower case letters. This will print out some 30 lines from EDIKIT1, all containing the string "ptr".

Of course, you can enter much longer strings to narrow down the search, including whole segments of text with embedded spaces or with spaces at beginning or end; hence the need for the delimiters, //. In fact, you may use any character as a delimiter; *FT.?ptr? or *FT."ptr" will work just as well as *FT./ptr/.

The command recognises the first non-space character following the command itself as the

current delimiter, and treats everything else it finds up to the second occurrence of that character as the target string. This facility to use any delimiter is important in case you should wish to find a string containing "/". If you forget to type a closing delimiter which matches the initial one, a warning message will be printed.

```
3570LDA #ipbuff  S256:STA pptr
3580LDA #ipbuff e256:STA pptr+1:RTS
3590.reinitbptr
3600LDY#3:LDA(bptr),Y
3610CLC:ADCbptr:STAbptr
3610CLC:ADCbptr:STAbptr
3620LDA#0:ADC bptr+1:STA bptr+1:RTS
3620LDA#0:ADC bptr+1:STA bptr+1:RTS
3840LDY#1:LDA(bptr),Y:STA hi
3850INY:LDA(bptr),Y:STA lo
4090JSR inittptr:LDY#0
4110LDA(tptr),Y:BMI tokenfnd
4190CLC:ADC tptr:STA tptr
4190CLC:ADC tptr:STA tptr
4200LDA#0:ADC tptr+1:STA tptr+1
4200LDA#0:ADC tptr+1:STA tptr+1
4240.inittptr
4250LDA#ttable  S256:STA tptr
4260LDA#ttable e256:STA tptr+1:RTS
4340LDA(bptr),Y:CMP#nl:BEQ dtexit
4430.movetoknloop LDA(tptr),Y
4480INY:LDA(bptr),Y:ASL A:ASL A
4490PHA:#&C0:INY: u(bptr),Y
4510 u(bptr),Y:STA hi:LDA#&FE
4560.quotelp INY:LDA(bptr),Y
Search complete
```

*Typical display from *FTEXT searching for 'ptr' in Edikit itself*

Any other EDIKIT command entered at this stage (apart from *FTEXT) will result in a message saying "Not implemented".

HOW IT WORKS

The program is in five sections represented by the procedures initvars, assemble1, tokentable, assemble2 and links. The function of initvars is self-explanatory. PROCassemble1 contains the code to set up a standard service ROM header and to respond appropriately to *HELP commands and other star commands unrecognised by the OS. A star command which can be matched with any of the list in comtable (lines 2450 onwards) will result in a jump to the appropriate code via a list of calls at the start of PROCassemble2.

The tokentable procedure copies the token table from the Basic ROM and incorporates it into EDIKIT. The table is necessary for detokenising lines of Basic; the original table cannot easily be read directly from the EDIKIT ROM as EDIKIT occupies a parallel area of memory to Basic, viz. &8000 to &BFFF.

The code for *FTEXT is in PROCassemble2 (findtext). After reading the target string into &600 (iptxtparam) and initialising pointers and vectors, scanprog is called. This points to each line of the Basic program in turn, calling scanfortextstr which detokenises it and places it in a buffer at &700. The buffer is then searched for the target string. If the string is encountered, the line is printed out (prntline). This line by line examination of the program is repeated until the end-of-Basic marker is found (&FF), when "End of program" is printed. Finally PROClinks provides a list of JMPs which will link EDIKIT1 with EDIKIT2. More of this in future articles.

```
  10 REM Program EDIKIT1
  20 REM Version B1.00
  30 REM Author  W. D. Hine
  40 REM BEEBUG  December 1989
  50 REM Program subject to copyright
  60 :
 100 MODE 7
 110 start=&8000:size=&C00:DIMcode &C40
 120 PROCinitvars:PROCassemble1
 130 ttadr%=&8071:REM BASIC II
 140 REM ttadr%=&8456:REM BASIC IV
 150 PROCtokentable(ttadr%)
 160 PROCassemble2:PROClinks
 170 PRINT"SAVE and LOAD ROM? Y/N"
 180 A=GET AND&DF:IFA<>ASC"Y"THENEND
 190 OSCLI("SAVE ediroml "+STR$~code+"+
"+STR$~(O%-code))
 200 OSCLI("SRLOAD ediroml 8000 X")
 210 PRINT"Press Ctrl-Break to initiali
se"
 220 END
 230 :
1000 DEF PROCinitvars
1010 ipbuff=&600:buffer=&700:page=&18
1020 top=&12:commandln=&F2:nl=&0D:cr=nl
1030 rem=&F4:def=&DD:quote=ASC""""
1040 space=ASC" ":query=ASC"?"
1050 star=ASC"*":numtoken=&8D
1060 :
1070 oswrch=&FFEE:osnewl=&FFE7
1080 osbyte=&FFF4:osword=&FFF1
```

```
1090 osrdch=&FFE0:osasci=&FFE3
1100 :
1110 flags=&70
1120 REM 0=print leading spaces
1130 REM 1=assembler opcode
1140 REM 2=find tokenised line no
1150 REM 6=first signif digit found
1160 REM 7=assembler on/off
1170 ay=&71:ex=&72:wy=&73:strlen=&74
1180 str2len=&75:token=&76:tokn=&77
1190 hitokn=&78:digits=&7B:REM &7B-&7F
1200 bptr=&80:tptr=&82:pptr=&84:lo=&86
1210 hi=&87:scantype=&88:action=&8A
1220 ipdelim=&8C
1230 ENDPROC
1240 :
1250 DEF PROCassemble1
1260 :
1270 FOR pass=4 TO7 STEP3
1280 P%=start:O%=code:[ OPT pass
1290 :
1300 .romheader
1310 EQUB0:EQUW0
1320 JMP servicertn:EQUB&82
1330 EQUBcopyrt MOD256:EQUB1
1340 .title
1350 EQUS"EDIKIT ":EQUB0
1360 EQUS"1.00":EQUB0
1370 .copyrt
1380 EQUB0:
1390 EQUS"(C) BEEBUG 1989":EQUB0
1400 :
1410 .servicertn
1420 CMP#9:BEQ helprtn
1430 CMP#4:BNE srexit:JMP commandrtn
1440 .srexit RTS
1450 :
1460 .helprtn
1470 PHA:TXA:PHA:TYA:PHA
1480 LDA(commandln),Y:CMP#nl
1490 BNE chcktitle:JSR pthelpmess
1500 LDX#255
1510 .prtsubhead
1520 INX:LDA subhead,X:BEQ subhdexit
1530 JSR osasci:JMP prtsubhead
1540 .subhdexit
1550 JMP helpexit
1560 .chcktitle
1570 LDX#255:DEY
1580 .chcktitlelp
1590 INX:INY
1600 LDA(commandln),Y:AND#&DF
1610 CMP title,X:BEQ chcktitlelp
1620 LDA title,X
1630 CMP#space:BEQ extendedhelp
1640 .helpexit
1650 PLA:TAY:PLA:TAX:PLA:RTS
1660 .pthelpmess
1670 JSR osnewl:LDX#&FF
1680 JSR pthlpmesslp:JSR pthlpmesslp
1690 JSR osnewl:RTS
1700 .pthlpmesslp
1710 INX:LDA title,X:BEQ hmexit
1720 JSR osasci:JMP pthlpmesslp
1730 .hmexit RTS
1740 .extendedhelp
1750 JSR pthelpmess:JSR osnewl:LDY#255
1760 LDA#comlist DIV256:STA hi
1770 LDA#comlist MOD256:STA lo
1780 .comlistlp
1790 INY:LDA(lo),Y:BMI exthlpexit
1800 JSR osasci:CPY#255:BNE comlistlp
1810 INC hi:JMP comlistlp
1820 .exthlpexit
1830 PLA:TAY:PLA:TAX:PLA
1840 LDX &F4:LDA#0:RTS
1850 .subhead
1860 EQUS"  Edikit":EQUWnl
1870 .comlist
1880 EQUS"  FBASIC <element>":EQUBnl
1890 EQUS"  FTEXT /<string>/":EQUBnl
1900 EQUS"  FPROCFN":EQUBnl
1910 EQUS"  LFROM <element>":EQUBnl
1920 EQUS"  LPROC <procname>":EQUBnl
1930 EQUS"  LFN <fnname>":EQUBnl
1940 EQUS"  RBASIC <element>/<element>"
1950 EQUBnl
1960 EQUS"  RTEXT /<string1>/<string2>/
"
1970 EQUBnl
1980 EQUS"  VARLIST [%]|[<init letter>]"
1990 EQUBnl
2000 EQUS"  SYSINF":EQUBnl
2010 EQUS"  FKDEFS":EQUBnl
2020 EQUS"  PCOMM":EQUBnl
2030 EQUS"  QCOMM":EQUBnl
2040 EQUS"  RCOMM":EQUBnl
2050 EQUS"  SCOMM":EQUBnl
2060 EQUS"  TCOMM":EQUBnl
2070 EQUS"  UCOMM":EQUBnl
2080 EQUS"  VCOMM":EQUBnl
2090 EQUS"  WCOMM":EQUBnl
2100 EQUS"  XCOMM":EQUBnl
2110 EQUS"  YCOMM":EQUBnl
2120 EQUS"  ZCOMM":EQUBnl
2130 EQUB&FF
2140 :
2150 .commandrtn
2160 TYA:PHA:LDX#0
2170 .testcom
2180 PLA:PHA:TAY:LDA(commandln),Y
2190 AND#&DF:CMPcomtable,X:BNE nxtcomlp
2200 .comlp
```

Continued on page 53

A MUG's Game

Tired of living on this mortal coil, with all that that entails. Cast off your sloth, surround yourself with a new persona, and enter the magic, mystical world of multi-user adventure games. Nicholas de la Valee has all the details.

WHAT IS A MUG?

A MUG is a Multi User Game; it is a text-only adventure game in which more than one person plays at a time. The people playing can 'see' each other and 'speak' to each other through the game. They can also steal items from each other, kill each other, help each other and so on.

Because of its multi-user nature, a MUG usually runs on a host computer system which you access via your own micro and a modem - be warned, you could end up with some nasty phone bills if you become too addicted to the role of adventurer.

The aim of a MUG, like most adventure games, is to gain points, increasing your status to reach the ultimate goal of immortality.

Once the highest levels are reached, some games make the player immortal, others just make him very powerful, while others just reveal yet another set of levels through which he still has to progress. A player may also be given certain powers over the other players.

To gain points in these games, you usually have to gather treasure and drop this in a certain place or sell it somewhere. For example, in Shades, once you have any treasure, you have to go and drop it in the Mad King's Room.

WHAT ARE MOBILES?

Mobiles are computer generated creatures, which roam about the adventurer's demesne. They have different characteristics; some of them attack on sight and others will never attack; some of them block exits and others defend treasure. Generally they are a pain, but points can be gained for killing them.

MAGIC, SPELLS AND THE LIKE

Just as in Dungeons and Dragons, the players have the ability to cast spells. As the players gain in status, the spells they cast become more and more powerful. At the level of Wizard, the player acquires a set of spells which allow him a certain degree of control over the land and the other players in the game.

ARCH-WIZARDS

In all the games, there are a certain number of players who manage the game. These people make sure that participants behave themselves, using such strong spells as BLOT and BAN. Recalcitrant players can be barred from all further participation in the game.

KILLING AND FIGHTING

In all the games, there is a certain amount of violence. Usually, during a fight, the person at the highest level wins, but players may also start hurling spells at each other, or stealing each other's weapons, making the outcome more doubtful.

A set of rules applies when one player 'kills' another. If the person starting the fight is killed he loses all his points and returns to the status of novice. If the innocent party is the one to lose his life, he still loses half his points. The winner of the fight always gains points for defeating another opponent, but becoming a mass murderer damages your image in the game, and you may find other players ganging up on you.

It usually takes some while to get used to a game, but once this has been achieved, players usually try to build an image for themselves around an assumed persona, and then act the part.

PROBLEMS

If you have any problems with access to any of these games, you can contact me (Nicolas de la Vallee) by any of the following means:

Prestel MBX : 011111177
Ariachus on the Mirrorworld mail System.
Alendil on the Mirrorworld mail System.
Message to the sysop on my board:
0342 712248 (viewdata).
Message to Nicholas de la Vallee on Byte Back:
01-959 8105 (scrolling).
Or send me a letter at Deepdale, 28 Calonne Road, Wimbledon, London SW19 5HJ.

A FULL LIST OF MUGS

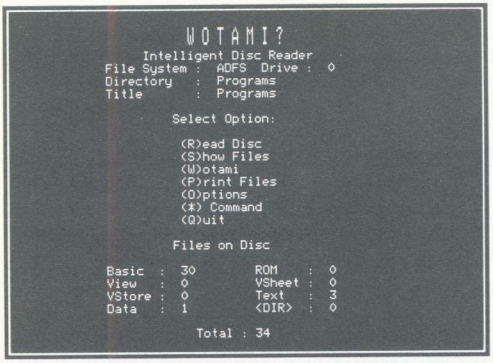
Here is a list giving details of all known multi-user games.

Mirrorworld	0883-844 044 0883-844 164
Times	24 hrs
Type of Reset	Autoset
Owner	Pippin
Baud	1200/75
Cost	Free!
Access	Type /MW as soon as you have got the introduction.
Gods	01-994 9119
Times	24 hrs Sat-Sun, evenings in the week.
Type of Reset	Global
Owner	Tiger Tiger
Baud	1200/75
Cost	£11.95 for 20 hrs playing time or 1 month unlimited.
Access	Guest account is GUEST; to access the game type 'G' from the main menu.
Quest 1	0883-844 044 0883-844 164
Times	24 hrs
Type of Reset	Autoset
Owner	Amstar
Baud	1200/75
Cost	Free!
Access	Type /QUEST as soon as you have got the introduction.
Realm II	(0272) 685485
Times	24 hrs (erratic)
Type of Reset	Global
Owner	Nanjusi
Baud	1200/75
Cost	Free!
Access	Just press Return a few times once you have got the carrier.
Wanderland	01-653 5246
Times	24 hrs
Type of Reset	Global
Owner	Wanda
Baud	1200/75
Cost	Free!
Access	Once the carrier is secure, press Return at five second intervals until you get a reaction!
Essex MUD	JANet Number, address is A000049600000; PSS Number, address is A220641141.
Times	2am to 7am weekdays and 2am to 10am Sat. and Sun.
Type of Reset	Global
Owner	Essex University
Baud	Depends on which JANet number.
Cost	Free!
Access	Once connected to the JANet computer, type 'HOST 1' at the prompt and then type: LOGIN 2653,2653 the password is guests. Once on, access to the game is obtained by typing MIST.
MUD II	01-583 1275 01-583 3000 01-583 1200
Times	All day Sat-Sun, Eves Weekdays
Type of Reset	Unknown
Owner	MUSE Ltd.
Baud	1200/75 300/300 1200/1200
Cost	£4.95 per 20 units
Access	Type 'CALL 41' at the PAD>prompt; from there, the USERNAME is MUDGUEST and the Password is PROSPECT.
Federation II	051-255 0225 (Martix BBs)
Times	24 hrs
Type of Reset	Unknown
Owner	Sysop
Baud	1200/75 & 300/300
Cost	Membership to the board.
Access	Through the board only if
you	are a member.
AMP	(0202) 678537 (300/300 Baud) PSS Address is A22020010700
Times	7pm to 6am every day, all day Sunday
Type of Reset	Global
Owner	Mike
Baud	300/300 on phone number
Cost	Free!
Access	The guest account isET2147.

Disc File Identifier (Part 2)

Alan Mothersole extends last month's program to cater for the ADFS, and to provide more detailed analysis of disc contents.

It is assumed that you have typed in and debugged the listing from Part 1 and saved it as *WOTdfs* without renumbering.



Using WOTAMI with ADFS

Two modules are presented here, and either one or both can be added to *WOTdfs* depending on your computer. If you don't have (or don't use) the ADFS, ignore this month's listing 1.

ADDING THE NEW FEATURES

Type in, if required, listing 1 with the line numbers exactly as shown and save as *WOTadfs*. Similarly, type in listing 2, again precisely maintaining the line numbers and save as *WOTwot*.

MERGING OF FILES

To merge the new ADFS module and last month's DFS program proceed as follows. Type:

```
NEW
LOAD"WOTadfs"
*SPOOL temp
LIST
*SPOOL

NEW
LOAD"WOTdfs"
*EXEC temp
SAVE"WOTfs"
```

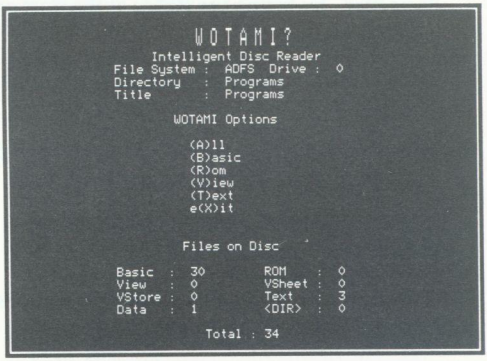
where *WOTfs* is a new program for both DFS and ADFS use (note: 'temp' is a temporary spooled file and can be deleted after use).

To merge the Wotami function (provided as listing 2 and saved as *WOTwot*) proceed in a similar way as described above to merge *WOTwot* and *WOTfs*, saving the final version as *WOTAMI*.

Obviously, if you don't require the ADFS compatibility, just merge *WOTdfs* and *WOTwot*.

OPERATION

Load and run the new program *WOTAMI*, and wait for the menu to be displayed. First check that the disc filing system set by the computer matches the disc to be read. If not use the <O> option to change it. Similarly set up the appropriate disc drive.



Using the WOTAMI feature

Pressing <R> will read the filenames and identify them. If the disc is DFS then all files on the selected side will be identified. However, for ADFS an option is given after the catalogue has been displayed to change the directory. Normal ADFS rules apply e.g. $,^ or directory name. Once the disc has been read a summary of the files will be shown beneath the menu.

There are two ways of displaying the information. <S> will show a list of the files together with their identification, length and the title or directory of the disc on which they are stored.

```
==================================
            BASIC FILES
==================================

Filename : BBCPOST
Load Add : FFFF0E00
Exec Add : FFFF802B
Length   : 3791 bytes
REMs :
 Program BBCPOST
 Version B 2.0
 Author  Willem van Schaik
 BEEBUG  December 1989
 Program subject to copyright
----------------------------------
```

Example printout using 'show' option from WOTAMI

Pressing <W> gives the Wotami menu to display details of:

1. Basic files - by listing the REMs at the start of a Basic program.
2. ROM Images - by displaying ROM header information.
3. View files by printing out the first line provided it had been saved with a CO stored command.
4. Text files by printing ASCII characters up to the first carriage return.

A hard copy can be obtained for both options by using <P> from the main menu.

MAIN PROCEDURES (ADDITIONAL)

PROCreadadfs - Read filenames from csd of ADFS disc.

PROCdir - If ADFS find which names are sub-directories.

PROCrBasic - Read REMs from Basic file.

PROCrROM - Read ROM header information from disc and sort into format.

PROCrView - Read Comment line of View file.

PROCrText - Read first line of text file.

PROCwotami - Gives more detailed information on ROM, Basic, View and Text files if present. Uses procedures PROCwBasic, PROCwROM, PROCwView and PROCwText.

NOTE: The magazine disc/tape contains the complete program combining this month's and last month's modules.

Listing 2

```
   10 REM Program WOTAMI
   20 REM Version 1.2
   25 REM >> ADFS module installed <<
   30 REM Author Alan Mothersole
   40 REM BEEBUG December 1989
   50 REM Program subject to copyright
   60 :
  110 ONERROR GOTO 10040
 1500 IF dfs THEN f$="DFS" ELSE f$="ADFS
"
 1750 IF dfs PROCdfscat:ENDPROC
 1755 PROCadfscat
 2365 PROCdir
 2421 IF (lock AND &2)=&2 THEN lock$(Z%)
=lock$(Z%)+"W"
 2422 IF (lock AND &1)=&1 THEN lock$(Z%)
=lock$(Z%)+"R"
 3155 IF A=102 PROCchfile
 3300 IF dfs=TRUE THEN OSCLI("DRIVE "+A$
) ELSE OSCLI("MOUNT "+A$)
 3440 IF dfs PRINT"DFS    ";tk%;" Tracks
" ELSE PRINT"ADFS"
 3460 IF dfs PRINT"Drive       : ";dr% E
LSE PRINT"Directory   : ";csd$
 3580 IF dfs dfree%=free%-dlen%
 4990 :
 5000 DEF PROCadfscat
 5010 OSCLI("CAT")
 5020 IF dfs=FALSE THEN PROCreadcsd
 5030 PRINT'"<DIR> (";csd$;") ";:INPUT D
ir$
 5040 IF Dir$="" THEN Dir$=csd$
 5050 IF Dir$<>csd$ THEN OSCLI("DIR "+Di
r$)
 5060 PROCcen(CHR$131+"Reading ADFS File
names")
 5070 PROCreadcsd:PROCdtitle:PROCreadadf
s
 5080 IF nof%=0 THEN ENDPROC
 5090 ENDPROC
 5100 :
 5110 DEF PROCreadadfs
 5120 LOCAL A%,X%,Y%,Z%
```

```
 5130 ?&70=dr%:X%=&70:Y%=0:A%=&71:CALL&F
FF1
 5140 dfree%=!&70
 5150 FOR X%=0 TO 550:buffer%?X%=0:NEXT
 5160 A%=&8:X%=block%MOD256:Y%=block%DIV
256
 5170 ?block%=0:block%!1=buffer%
 5180 block%!5=47:block%!9=0
 5190 CALL &FFD1:Z%=0
 5200 REPEAT:name$(Z%)=""
 5210 FOR Y%=1 TO namax%
 5220 IF buffer%?(Y%+(namax%+1)*Z%)>0 na
me$(Z%)=name$(Z%)+CHR$(buffer%?(Y%+(nama
x%+1)*Z%))
 5230 NEXT:Z%=Z%+1
 5240 UNTIL Z%=fmax% OR LENname$(Z%-1)=0
 5250 nof%=Z%-1
 5260 ENDPROC
 5270 :
 5280 DEF PROCdir
 5290 IF dfs ENDPROC
 5300 IF (USR(&FFDD) AND &FF)=2 di%=di%+
1:type%(I%)=8
 5310 ENDPROC
 5320 :
 5340 DEF PROCchfile
 5350 LOCAL A
 5360 PRINT':PROCcen("(D)FS "):PROCcen("
(A)DFS")
 5370 REPEAT:A=GET:A=A OR &20:UNTIL A=10
0 OR A=97
 5380 IF A=100 OSCLI("DISC")
 5390 IF A=97 OSCLI("FADFS"):PROCchdrv
 5400 read=FALSE
 5410 ENDPROC
 5420 :
10020 DATA 3,(D)isk Drive,(F)iling Syste
m,e(X)it to Menu
10040 IF ERR=214:VDU7:PRINT"Directory no
t found":A=GET:RUN
10045 IF ERR=200 PROCcen(CHR$131+"DISK C
HANGED"):OSCLI("MOUNT "+STR$(dr%)):RUN
```

Listing 2

```
   10 REM Program WOTAMI
   20 REM Version 1.2
   28 REM >> Wotami module installed <<
   30 REM Author Alan Mothersole
   40 REM BEEBUG December 1989
   50 REM Program subject to copyright
   60 :
 1385 IF A%=119 CLS:PROCwotami
 4020 IF A=115 PROCshow ELSE PROCwotami
```

```
 6000 DEF PROCrBasic
 6010 IF type%(I%)<>1 THEN ENDPROC
 6020 C%=OPENUP(name$(I%))
 6030 rem%=1
 6040 finish=FALSE:exit=FALSE:found=FALS
E
 6050 PRINT'"Filename : ";name$(I%)
 6060 PRINT"Load Add : ";~load(I%)
 6070 PRINT"Exec Add : ";~ex(I%)
 6080 PRINT"Length   : ";length(I%);" by
tes"
 6090 PRINT'"REMs :"
 6100 REPEAT
 6110 data=BGET#C%
 6120 IF data=&F4 THEN found=TRUE
 6130 IF PTR#C%=100 OR EOF#C% THEN exit=
TRUE
 6140 UNTIL found OR exit
 6150 IF exit PRINT">> NO REMS IN FILE <
<":PRINT:PROCline:CLOSE#C%:ENDPROC
 6160 REPEAT
 6170 m$="":PROCdat:PRINTm$
 6180 PTR#C%=PTR#C%+3
 6190 data=BGET#C%
 6200 IF data<>&F4 THEN finish=TRUE
 6210 UNTIL finish=TRUE
 6220 PRINT:PROCline
 6230 CLOSE#C%
 6240 ENDPROC
 6250 :
 6260 DEF PROCdat
 6270 REPEAT
 6280 data=BGET#C%
 6290 IF (data<&20 AND data<>&0D)OR(data
>&7E AND data<>&FF) THEN data=&20
 6300 m$=m$+CHR$(data)
 6310 UNTIL data=&0D OR data=&FF
 6320 ENDPROC
 6330 :
 6340 DEF PROCrROM
 6350 IF type%(I%)<>2 ENDPROC
 6360 rname$="":copy$="":rtype$="Service
"
 6370 C%=OPENUP(name$(I%))
 6380 data=BGET#C%
 6390 IF data=&4C THEN rtype$="Language"
 6400 PTR#C%=PTR#C%+7:Ivs=BGET#C%
 6410 rname$=FNgdata:vs$=FNgdata
 6420 PTR#C%=0
 6430 REPEAT:data=BGET#C%:UNTIL data=&28
 6440 data=BGET#C%
 6450 IF data<>&43 THEN CLOSE#0:ENDPROC
 6460 PTR#C%=PTR#C%-2
 6470 copy$=FNgdata
```

```
 6480 IF vs$=copy$ THEN vs$=""
 6490 IF vs$="" THEN vs$=FNgdata
 6500 CLOSE#C%
 6510 PRINT"ROM name      ";rname$;'"ROM
type     ";rtype$
 6520 PRINT"Internal No. ";Ivs;'"Length
     ";~length(I%)
 6530 PRINT"Version       ";vs$;'"Copyrig
ht     ";copy$
 6540 PRINT:PROCline
 6550 ENDPROC
 6560 :
 6570 DEF PROCrView
 6580 IF type%(I%)<>3 ENDPROC
 6590 co$=""
 6600 C%=OPENUP(name$(I%))
 6610 REPEAT:data=BGET#C%:UNTIL data=&43
 6620 PTR#C%=PTR#C%+1
 6630 REPEAT:data=BGET#C%
 6640 co$=co$+CHR$(data)
 6650 UNTIL data=&0D
 6660 CLOSE#C%
 6670 PRINT"Filename : ";name$(I%);TAB(2
4)"Length : ";length(I%)
 6680 PRINT'"COMMENT :":PRINTco$:PRINT:P
ROCline:PRINT
 6690 ENDPROC
 6700 :
 6710 DEF PROCrText
 6720 IF type%(I%)<>6 ENDPROC
 6730 m$=""
 6740 C%=OPENUP(name$(I%))
 6750 PROCdat:CLOSE#C%
 6760 PRINT"Filename : ";name$(I%);TAB(2
4)"Length : ";length(I%)" bytes"
 6770 PRINT'"DATA :":PRINTm$:PRINT:PROCl
ine:PRINT
 6780 ENDPROC
 6790 :
 6800 DEF PROCwotami
 6810 LOCAL A,T%,exit
 6820 IF read=FALSE ENDPROC
 6830 exit=FALSE
 6840 REPEAT
 6850 LOCAL A
 6860 PROCtitle:PRINT
 6870 PROCcen(CHR$134+"WOTAMI Options")
 6880 PRINT:RESTORE 10035
 6890 FOR T%=1TO6:READ A$
 6900 PRINTTAB(12)CHR$130;A$
 6910 NEXT:PROCupdate
 6920 REPEAT
 6930 A=GET:A=A OR &20
 6940 UNTILA=97 ORA=98 ORA=114 OR A=116
ORA=118 ORA=120
 6950 CLS
 6960 IF prt THEN VDU2
 6970 IF A=120 exit=TRUE
 6980 IF A=98 PROCwBasic
 6990 IF A=114 PROCwROM
 7000 IF A=118 PROCwView
 7010 IF A=116 PROCwText
 7020 IF A=97 PROCwBasic:PRINT':PROCwROM
:PRINT':PROCwView:PRINT':PROCwText
 7030 VDU3
 7040 IF NOT exit THEN PRINT':PROCky
 7050 :
 7060 UNTIL exit
 7070 ENDPROC
 7080 :
 7090 DEF PROCwBasic
 7100 IF ba%=0 THEN ENDPROC
 7110 PROCline2:PRINTTAB(15)"BASIC FILES
"
 7120 PROCline2:PRINT
 7130 FOR I%=0 TO nof%-1:PROCrBasic:NEXT
 7140 ENDPROC
 7150 :
 7160 DEF PROCwROM
 7170 IF ro%=0 THEN ENDPROC
 7180 PROCline2:PRINTTAB(15)"ROM IMAGES"
 7190 PROCline2:PRINT
 7200 FOR I%=0 TO nof%-1:PROCrROM:NEXT
 7210 ENDPROC
 7220 :
 7230 DEF PROCwView
 7240 IF vi%=0 THEN ENDPROC
 7250 PROCline2:PRINTTAB(15)"VIEW FILES"
 7260 PROCline2:PRINT
 7270 FOR I%=0 TO nof%-1:PROCrView:NEXT
 7280 ENDPROC
 7290 :
 7300 DEF PROCwText
 7310 IF tx%=0 THEN ENDPROC
 7320 PROCline2:PRINTTAB(15)"TEXT FILES"
 7330 PROCline2:PRINT
 7340 FOR I%=0 TO nof%-1:PROCrText:NEXT
 7350 ENDPROC
 7360 :
 7370 DEFPROCline2:PROCcen(STRING$(36,"=
")):ENDPROC
 7380 :
10010 DATA 7,(R)ead Disc,(S)how Files,(W
)otami,(P)rint Files,(O)ptions,(*) Comma
nd,(Q)uit
10030 DATA 3,(S)how,(W)otami,e(X)it
10035 DATA (A)ll,(B)asic,(R)om,(V)iew,(T
)ext,e(X)it
```

Adventure Games

by Mitch

Product	**Stranded**
Supplier	**Robico Software** **3 Fairland Close, Llantrisant,** **Mid Glamorgan CF7 8QH.** **Tel. (0443) 227354**
Price	**£17.95 inc. VAT** **(available from BEEBUG)**

Somewhere, in a galaxy far, far away, a young hero smuggles himself aboard a cargo freighter heading for adventure and the stars. What a pity he picked the pantry to hide in. By tea-time he was wrapped in chains and waiting for summary execution - that's where you come in!

Escaping from your chains is an easy affair, even crashing the ship onto a nearby planet will present few problems. But how do you now propose to solve the challenge that you've crashed slam-bang into the middle of an ancient, gladiatorial slave fortress? The planet is governed by Baron Knutah (Nutter - geddit?) and the fortress is amid Lotsatrees Forest. Your only avenue of escape appears to be via promotion out of the slave quarters, and into the soldiery of the Baron.

Using fair means and foul (and some pretty hilarious), you can soon be moving freely around the landscape, picking up a seemingly endless supply of bits and pieces. After all my years of adventuring, I still laugh when I visualise my computerised alter-ego climbing the side of a wall; supposedly wearing a gas-mask and carrying 'umpteen' items, (including a mattress, axe and a portrait) which I forgot I was still carrying.

Old style adventures tended to be unpopulated places - as handling conversations between the player and computerised characters was too tricky. Gradually, immobile personalities were added to bring extra realism. Current games now tend to include animated characters who must be followed and then bribed, cajoled or thumped into assisting you. *Stranded* includes a number of these obstinate personalities. There are the two soldiers Cowmuck and Dunkov, Yarvik the Border Guard, and Vardan the Centurion, all of whose 'hash' you must settle before freedom is gained. Unlike the humble Beeb, the Archimedes version of the game includes graphic screens, but as with all adventures, the game must inevitably live or die by the standard of the writing. Happily this one lives up to the task admirably, with witticisms that are both neat and to the point.

With approx 140 items of interest, plus 150 meaningful locations (as opposed to the hundreds of dummy locations found in some games), this space romp is 'lotsa' fun and 'cheeky-with-it'. I particularly liked such items as the 'Acme Box of Anger' which is useful when you want to let off steam.

The problems don't appear to be mind-boggling - at least not up to the score of 200/800 where I'm currently thrashing around. So it is Lord Fun, rather than the Dark Lord, who co-rules this crazy planet with Baron Knuttah. The game appears to enjoy watching your attempts to struggle with a problem and pokes fun at your efforts. Successive replies such as: 'No, that's not it', followed by 'Neither is that!', lift the level of enjoyment.

Robico Software has a good track-record with BBC adventures, and it's nice to see that it continue to produce worthwhile games. *Stranded* was written by Tony Heap instead of the eponymous Rob O'Leary, who now manages Robico. I don't believe I've played any other games by Tony, but if this game succeeds as well as it deserves to, then I'm sure we'll be crossing photon-emitters again.

A FIND Utility

Bernard Hill describes a simple yet highly effective utility with a multitude of applications.

On the BEEBUG magazine disc for Vol.8 No.6 there appeared an MS-DOS utility called FIND. This utility is normally found on MS-DOS system discs but had not been implemented for DOS+ on the Master. The utility is often underestimated in its usefulness - so much so that I have produced one for the standard BBC micro.

Quite simply, FIND locates a particular string within any text file, and displays the complete line on which it occurs. Since it is a machine code program it can be called with a star command, and works equally well from within the View command screen, for example, as from Basic.

Once the program has been typed in (and saved as, say, SFIND), it may be run. It produces and saves to disc a utility called FIND which may be called up in the following ways:

```
*FIND
```

gives a brief reminder of the correct syntax, while:

```
*FIND <filename> <string>
```

lists to the screen every complete line in the specified file which contains the given string.

For the purposes of matching strings, case is ignored so that "Fred" matches with "FRED", "fred", etc. Around '<string>' quotation marks (") are optional so that spaces could be included if required. Should any non-printable characters be detected in the file then they are replaced by "?" on the subsequent display. There is also a form:

```
*FIND <filename> <string> N
```

which is identical to the above except that line numbers preface the lines displayed.

APPLICATIONS

Clearly, the utility can be called from within a word processor to check on the contextual use of any given word, but I find the principal use of FIND within a simple database (but still using a word processor). As an example let us consider a record collection. The records are produced with a word processor (e.g. Wordwise, Edit or View), and arranged loosely tabulated one-to-a-line with title and performer such as:

```
Bach        Brandenburg Concertos      Solti, LSO
Mozart      Symphonies 40,41           Previn, LSO
Pink Floyd          Dark side of the moon
Mozart      Piano Concertos 23 & 24    Brendl, VPO
Pink Floyd          The Wall
...
```

To each of these entries could be added, of course, anything else of interest such as medium (e.g. cassette, CD), date, or a location. Once this file - say called "RECORDS" - has been created, then it could be searched as follows:

```
*FIND RECORDS MOZART
*FIND RECORDS "PINK FLOYD"
*FIND RECORDS CONC
*FIND RECORDS PREVIN
```

Clearly this may be used to locate any particular value of any particular field, and can be used a number of times as required: suppose we want to locate all Beethoven Symphonies, then a spool file can be used:

```
*SPOOL TEMP
*FIND RECORDS BEETHOVEN
*SPOOL
*FIND TEMP SYMPHON
```

Note the short form to catch both 'Symphony' and 'Symphonies'.

The utility is equally at home in a Basic environment. Suppose you wish to locate all references within a program to the variable 'start':

```
*SPOOL TEMP
LIST
*SPOOL
*FIND TEMP START
```

You could actually use the saved Basic program itself, but beware that all keywords are tokenised and will appear as '?', and line counts (via the 'N' parameter) could be wrong due to the occurrence of the byte &0D in the saved program.

LIMITATIONS AND CUSTOMISATION

As always with the BBC micro, the question of where to place these utilities arises. The utility is just under &200 bytes and I have used page 9, but on a Master line 100 should probably be replaced with R%=&DD00 as this area is reserved for this type of utility. If you have a standard DFS you may get away with using R%=&1700 but notice also that the program uses &A0 bytes starting at location &680. This seems to have no effect on View 3, but test out with your own word-processor, and use another location by changing the value of S% in line 100 accordingly.

Note that lines of text over 128 bytes are truncated to 128 bytes, and that search patterns should be limited to 32 characters.

Databases which use Basic's PRINT# and INPUT# instructions store text strings in reverse character order, without Return or Line Feed as a delimiter. The FIND utility is, on the whole, unsuitable for locating a string in such a file, but I hope to deal with this in a separate program in a later issue.

```
 10 REM Program Find
 20 REM Version B1.0
 30 REM Author  Bernard Hill
 40 REM BEEBUG  December 1989
 50 REM Program subject to copyright
 60 :
100 R%=&900:S%=&660
110 line=S%+&20:match=S%
120 filename=line:MODE7:HIMEM=&7000
130 FOR opt=4 TO 6 STEP 2
140 P%=R%:O%=HIMEM
150 [ OPT opt
160 JSR skipsp:CMP #13:BNE ov0
170 .help LDX #0:.loop LDA helpt,X
180 BEQ endhelp:JSR &FFE3
190 INX:BNE loop:.endhelp RTS
200 .ov0 LDX#0:STX quotes:STX N
210 STX Lcount:STX Lcount+1
220 .loop STA filename,X:INX:INY
230 LDA (&F2),Y:CMP #13:BEQ help
240 CMP #32:BNE loop
250 LDA #13:STA filename,X:JSR skipsp
260 CMP #ASC"""":BNE ov1:DEC quotes
270 INY:LDA (&F2),Y
280 CMP #ASC"""":BNE ov1
290 LDA #0:STA match:INY:JMP next1
300 .ov1 LDX #0
310 .loop JSR uc:STA match,X:INX:INY
320 LDA (&F2),Y:JSR uc
330 CMP #13:BEQ finstr
340 BIT quotes:BMI ov2
350 CMP #32:BEQ next1:BNE loop
360 .ov2 CMP #ASC"""":BNE loop:INY
370 .next1:JSR skipsp:CMP #13
380 BEQ finstr:AND #&DF:CMP #ASC"N"
390 BEQ setN:.badstring
400 BRK:EQUB 0:EQUS "Bad String":BRK
410 .setN DEC N
420 .finstr LDA #0:STA match,X
430 LDA #&40:LDX #filename MOD 256
440 LDY #filename DIV 256
450 JSR &FFCE:CMP #0:BEQ nofile:TAY
460 .newline:LDX #0:.nextch:JSR &FFD7
470 BCS eof:CMP #13:BEQ eol
480 STA line,X:INX:CPX #129:BNE ov5
490 DEX:.ov5 BIT &FF:BPL ov6
500 LDA #0:JSR &FFCE
510 BRK:EQUB 17:EQUS "Escape":EQUB 0
520 .ov6:JMP nextch
530 .eol:INC Lcount:BNE ov8
540 INC Lcount+1:.ov8:LDA #0
550 STA line,X:JSR Qmatch:JMP newline
560 .eof LDA #0:JSR &FFCE:RTS
570 .nofile
580 BRK:BRK:EQUS "File not found":BRK
590 .skipsp:LDA (&F2),Y:CMP #32
600 BNE ret:INY:JMP skipsp:.ret RTS
610 .uc:CMP #ASC"a":BCC retn
620 CMP #ASC"z"+1:BCS retn:AND #&DF
630 .retn RTS
640 .Qmatch:TYA:PHA:LDX #0:STX yes
650 .starty:LDY #0:STX x
660 .next:LDA line,X:BEQ eoline
670 JSR uc:CMP match,Y:BNE restart
680 INX:INY:JMP next
690 .restart LDA match,Y:BEQ perfect
700 LDX x:INX:JMP starty
710 .eoline LDA match,Y:BNE over3
720 .perfect DEC yes:.over3:BIT yes
730 BPL retrn:BIT N:BPL ov9:LDX #0
740 LDA Lcount:STA work
750 LDA Lcount+1:STA work+1
760 .loop LDA #48:STA num:.loop2 SEC
770 LDA work:SBC tens,X:STA work
780 LDA work+1:SBC tens+1,X
790 STA work+1:BMI ov7:INC num
800 JMP loop2:.ov7 CLC
810 LDA work:ADC tens,X:STA work
820 LDA work+1:ADC tens+1,X
830 STA work+1:LDA num:JSR &FFEE
840 INX:INX:CPX #10:BNE loop
850 LDA #ASC":":JSR &FFEE
860 .ov9:LDX #0:.lp:LDA line,X
870 BEQ endln:CMP #31:BCC low
880 CMP #127:BCC ok
890 .low LDA #ASC"?":.ok JSR &FFEE
900 INX:JMP lp
910 .endln JSR &FFE7
920 .retrn:PLA:TAY:RTS
930 .x       EQUB 0
940 .N       EQUB 0
950 .Lcount  EQUW 0
960 .work    EQUW 0
970 .yes     EQUB 0
980 .quotes  EQUB 0
990 .num     EQUB 0
1000 .tens   EQUW 10000
1010         EQUW 1000
1020         EQUW 100
1030         EQUW 10
1040         EQUW 1
1050 .helpt EQUS "Syntax:"
1060 EQUW &D0D
1070 EQUS "*FIND <filename> <string> [N
]"
1080 EQUW 13
1090 ]
1100 NEXT
1110 c$="SAVE FIND "+STR$~HIMEM+" "+STR
$~O%+" "+STR$~R%+" "+STR$~R%
1120 PRINTc$:OSCLIc$
```

The Account Book, and The Invoice Program

Reviewed by John Woodthorpe

Products	**The Account Book (V3), and The Invoice Program**
Supplier	**Apricote Studios 2 Purls Bridge Farm, Manea, Cambs. PE15 0ND. Tel. (035 478) 432**
Price	**£27.95 each, or £49.95 for both, inc. VAT, p&p.**

Available in all disc formats, this software will work on all Acorn/BBC machines, except for the Electron (no mode 7), and is tube compatible. The additional requirements are an Epson compatible printer, a monitor (monochrome or colour), and any disc drive, other than one single sided. The packaging is fairly basic - just the disc(s) and manual in a plastic folder. Whatever programs you buy, the same manual is supplied, so that if you buy the Account Book, you can also read about the Invoice Program. The manual itself consists of 42 pages, 28 of which deal with the Account Book. Currently, it is photocopied from a rather dotty original, but a printed version is promised shortly.

THE ACCOUNT BOOK

An earlier version of this was reviewed in BEEBUG Vol.7 No.5 p.54, so I will just summarise its features for the benefit of newcomers, and mention the modifications. Any small business needs to keep records of income and expenditure. It helps if the recording process allows you to check bank statements, and keep a track of any cash-in-hand. Account Book performs all this and more. After making a working copy of the disc, and booting up, the program asks what number and type of disc drives you want to use. I have one double sided drive, and replied accordingly. Various files were then created on drive two (the other side of drive 0), a somewhat lengthy, but once only, process.

Let me say here that the approach of both manual and software is to make the whole installation and operation process as transparent as possible to the user - the manual gives the commands to format a blank disc and backup the original to create the working copy. Just in case you are unsure, it even tells you how to boot the disc, and suggests adjusting the contrast and brightness on a monochrome monitor until you can see the copyright message. This shows the thought that has gone into both packages - you don't need technical knowledge to use the software. These are programs for any small business, not just one selling computers!

Sales and payments are recorded in an easy-to-understand way, and any money received or invoiced is entered in the sales book as paid or unpaid. Any payments made (including money paid into the bank) or bills received, are recorded in the payments book. At the end of your financial year, all paid items are then printed out, together with any other relevant reports, ready for your accountant. It can also produce detailed sales and payment reports, trial balances, and profit and loss graphs.

The modifications made since the last review were mostly suggested by Apricote's customers, and widen the areas of usefulness of the program. They include more on-screen prompting for VAT entries, extended reporting facilities, covering searching for cheque and invoice numbers, receipts, dates, wildcard searching and sales label totalling. In addition, it is possible to reconcile payments and receipts with bank statements. In seven years as a Church Treasurer, I struggled with a combination of ViewSheet and a commercial program. I have now found the software I needed to do the job simply and easily! Indeed I understand that Church Treasurers, and Doctors, use this program, as well as more conventional businesses. The main limitation is that it only deals with one bank account. Perhaps provision for a second account might find its way into a future version?

THE INVOICE PROGRAM

Upon booting up this disc, the first question asked is whether you will be using it in conjunction with the Account Book. A positive

answer produces a prompt to change discs whilst data is read across. You don't have to use the two programs together, they can stand alone, but they work as an integrated package - invoices automatically being entered into the Account Book once you have printed them out using the Invoice Program. With two double-sided drives, keep the Invoice Program in drive zero, and the Account Book in drive one.

```
Date: 08.11.89                                  Invoice Num: 123456

Description                                     Price Ex/Vat  V%

Pre-packed spaghetti 2lb bags: @ £1.99 x 10          19.90  15
Baked Beans: Smiths own label £3.15 per case x 20 cases  63.00  15
Sausages: £0.89p per lb x 50lbs                      44.50  15
Wondercrisps: £2.99 per 24 x 10 cases : Salt & Vinegar  29.90  15
Smiths own label lozenges: £5.99 per pack x 200 packs  1198.00  15
A4 copypaper: 500 sheet boxes @ £2.59 x 6.5 boxes    16.83  0

(all the above have been input using the presets, with only
the relevant quantities added to complete the lines. The
program has worked out the totals automatically. Different
vat codes can also be used as you can see. And you can even
write explanations like this one! with a price at the end
if you like.....)                                    10.00  0
```

A sample invoice produced with this package

Once set up, the look and feel are those of the Account Book, and the first job is to enter your company details into the utilities menu. If you don't enter a VAT registration number, the program makes the logical assumption that you are not VAT registered, and modifies the printouts accordingly. If (when!?) the VAT rates (15%, 0%, and exempt) change, you can modify them.

The next thing to do is to set up the first invoice number - handy if you've been in business for two years before buying the program - and any pre- or post-fixes you want to use. After that, back to the main menu, press Return to select the Invoices Menu, move the cursor down to Stock Presets, and enter up to 100 stock lines or standard messages (see figure for examples).

The square brackets, [and], signify a calculated total. This can be called up later and incorporated into an invoice, leaving only the number of items to be added, along with the closing bracket]. When printed out, the correct total will appear for the number of items sold. This number of items need not be an integer - useful for those selling half a tonne of wheat, or 2.5 hours of music tuition.

Other facilities include entering standard messages which appear on the invoices, e.g. the names of directors and advertising slogans. In addition, overdue messages can be entered and automatically printed on statements prepared subsequently. The preset intervals are 28 days apart, but these can be changed if desired. As if all this were not enough, there's a 700 entry database of customers names, addresses, and other details.

Producing invoices and statements is very simple, and satisfying - a really professional job can be done, as shown by the example included. All the details can be standardised, or tailored to individual customers; you can even tell the software the code your printer uses for the pound sign - although this feature only applies to invoices, on other printouts it gives CHR$(96). The cure for this is to set the printer to the UK character set, and type #'s.

On the topic of criticisms, it really is hard to find any major ones. After printing the customer database, the printer is left in condensed mode, and has to be reset manually. Oh yes, and the contrast between the pretty mode 7 colours used for the menus, and the mode 3 black and white screens used for data entry is a bit severe. Whatever you could want, this software does it, even printing labels or envelopes for posting off your invoices!

CONCLUSION

As you may have gathered, I am very impressed with these programs. I must also mention the excellent after-sales service given by Mr. Pain of Apricote. It includes free telephone help, and minor software modifications. More substantial changes are charged for at a nominal rate, and updates offered to existing customers for £3 per disc. In addition, a version for business studies classes is shortly to be released. I think the best testimony to these programs is that Mr. Pain uses them for his own business. In the words of his advertisement, if you buy these packages, "you will not be disappointed".

BEEBUG Education

by Mark Sealey

INTRODUCTION

New educational programs for the BBC are becoming increasingly rare. It is interesting that most of those which are published are of relatively high quality. This month BEEBUG Education looks at three offerings for younger children all from software stalwarts, all three reviewed run on the BBC model B, Master and Master Compact, though state 3.5" or 5.25" disc when ordering. All three packages are offered by ***BBC Soft, P.O. Box 234, Wetherby, West Yorkshire LS23 7EU.*** Post and packing is an extra £1.50 per order.

There is an interesting comparison to be made between then and now - almost seven years since the first Acorn educational software began to appear. These latest titles have a sophisticated feel to them, employ some of the best graphics of which the 8-bit machine is capable, and assume a maturity in their users in terms of conventions of the screen and progress through the program. This would have been much less likely in the mid 80s, and yet at the end of a turbulent decade for education, in no way detracts from the usability of these products.

MAKE A WILDLIFE GARDEN

(£19.95 inc. VAT)

This typifies several welcome trends in good educational software. Perhaps chief among them is the sort of thematic approach that means that a high proportion of the supporting material referred to, and contained in the documentation, is concerned with off-computer activity.

This is a well-designed and presented package released to support the schools radio series Science Naturally for upper Juniors. Formerly known as Looking at Nature, it takes an approach to learning based on acquiring and working with scientific skills and values, rather than the 'absorption' of facts in, and by themselves.

So the pupil is responsible for designing a garden in which not only plants but also wild animals, minibeasts and insects flourish. The main graphic is a rectangular garden, which is viewed from one corner at a height of about two metres. "Features", shrubs, logs, a pond etc., can be added and moved at will.

The software is flexible and previous designs can be loaded and saved. There are - of course - clear instructions for using a data disc, transferring to a network and customising the software to the computer: number and types of disc-drives, colour printer, use of a mouse etc.

It is assumed that four factors affect the ecology of the garden: habitats themselves, the suitability of plants and animals in "co-habitation", rainfall and sunshine/shade.

These variables can be set and changed easily and graphically by the children, though they will need guidance through this part of the manual as it was obviously not written with their likely reading abilities in mind.

Then the mini-simulation is run - with reports optionally sent to printer - for one of three weather conditions and the fun starts. Sun loving plants sited in the shadow of the tree or a woodlouse located too far from its log will fail to flourish. Why?

It is in this interpretation of results that the real value of the package lies. There is a "results table", which can be photocopied and used for each run. So no insects or cherished plants will really die, and the children could end up with a plan for a garden which has both been conceived and fully explored without the heartache of horticultural disaster or the tedium of having to wait for plants actually to grow and suffer from the pupils' inexperience.

This is one of the best uses of the computer and executed very well in this instance. There is much additional information and source of

stimulus in the forty-odd page booklet that comes with it, including ten references for material for the age-group and a list of suppliers of wildlife. Then there is just the right amount of information on plant families - though some illustrations would have been handy. Again it will be necessary for a good reader to work at this with the children.

Lastly, Making a Wildlife Garden will clearly stimulate work away from the computer around such themes as food chains, plant identification and patterns of ecological regeneration and so on. As usual with BBC Software, there are lots of suggestions accompanying the package. In conclusion, this new arrival does everything that it sets out to do in a user-friendly manner, with non-intrusive sound effects and just the right blend of suggested work off the keyboard with child-centred decision making and interpretation at it. To be recommended.

THROUGH THE DRAGON'S EYE STORYBOARD AND ADVENTURE

(each £19.95 inc. VAT)

You learn, from the material accompanying the second of this months review titles that "The land of Pelamar is in danger. Its life force, the Veetacore, has exploded and without it, all colour and life is draining out of the land".

Also linked to a popular BBC Schools TV series, Look and Read, these programs consist of a simple publishing package, where users (of Primary, Lower Secondary age as well - perhaps - as in some Special Needs and Adult Literacy situations) can create, edit and save pages on the theme of the contemporary quest which forms the adventure unit of this software.

As a formula, it is not new (several titles from Resource tackle a similar task, for instance). There is a bank of 48 library graphics, of which 11 may appear at a time on any one page. The letters are double height, easy to move and manipulate, and the pages can be printed out. Particularly well thought out is the undo/restore facility. Indeed much of what is contained in the software has clearly been conceived with young users in mind, and anticipates many of the likely pitfalls to which they will be prone.

The Storyboard is well supported with ideas, sources and references in the documentation, and provides an excellent introduction to quest literature in general and the exploits of the Veetons and Widgets in particular.

They really start to come alive in the Adventure itself. Here too, there is much in the manual to fill out what the pupils will undoubtedly enjoy at the computer.

Again, operation of the software is easy and intuitive. This is an adventure with some thirteen segments, playing upon mathematical, spatial, linguistic, logical and visual skills and experience. It could be argued that some of the games are a little old-hat and based on less than sound educational principles (like pure phonics). Some children will just take to them; others may find them well beyond their abilities and become frustrated. It is assumed that these are all too easy for an adult for any kind of crib to be necessary!

Undoubtedly the design of the games will appeal to the target age-range, and contribute to the overall feel of a purposeful adventure. There is plenty of scope for follow up, and even preparatory maths and language work, though without the complexity or off-putting length of "serious" adventures such as L, for instance. One serious omission is a facility to save a partially completed game; though each can be played from a menu as well as consecutively.

Despite being in some senses the weakest of the three products under review this month, BBC Soft have here produced a set of programs which will provide some purposeful, stimulating and worthwhile new activities and link them to others (such as the TV series), with which the children are familiar. It is interesting to speculate for how long software producers can and will continue to come up with material of this quality. B

1st course

Using Operating System Routines (2)

This month, Mike Williams investigates some further operating system routines for the benefit of Basic programmers.

Last month I explained, in I hope adequately simple terms, some of the reasons why it is useful to know how to call operating system routines from within Basic, and the means to do this. To recap, Basic provides two keywords for this purpose, CALL which has some similarities with a procedure call, and USR which returns a value as does a Basic function. In either case we follow the keyword with the address of the routine we want to use.

Usefully, Basic automatically takes the values previously assigned to A%, X% or Y% as the values for the 6502 processor's accumulator (A), X and Y registers, which makes life easy. The USR function also returns the values of the A, X and Y registers as a single four-byte value, and I explained last time how the individual bytes may be separated if required. Please refer to that article where necessary, as I shall this month assume a knowledge of that aspect of the use of USR without further explanation.

What I want to do this month is to explore some of the more useful or interesting routines which we can access in this way. Sometimes this is an alternative to the FX call in Basic, sometimes (particularly where a value of some kind is to be returned) it is the only way in which this can be achieved. In all cases the approach is intended to help the more moderate of Basic programmers expand the range of functions which can be handled in even quite simple Basic programs.

First of all let us look at OSBYTE 138, which allows a specified character to be placed into the keyboard input buffer as though it had been typed from the keyboard. On entry, X% should be set to 0, and Y% to the ASCII value of the character to be inserted. As with every OSBYTE call, A% is set to the value which identifies this particular call (OSBYTE 138). Thus:

```
OSBYTE=&FFF4
X%=0
Y%=ASC(A$)
A%=138
CALL OSBYTE
```

where A$ contains the character to be used. One application of this call allows a program to simulate the actions of a human user when responding to a program.

We could also incorporate this in a procedure which allows a complete string of characters to be placed into the keyboard input buffer:

```
1000 DEF PROCkeybd_input(msg$,RT%)
1010 LOCAL A%,I%,X%,Y%
1020 OSBYTE=&FFF4:A%=138:X%=0
1030 IF RT% THEN msg$=msg$+CHR$13
1040 FOR I%=1 TO LEN(msg$)
1050 Y%=ASC(MID$(msg$,I%,1))
1060 CALL OSBYTE
1070 NEXT I%
1080 ENDPROC
```

Often, what we type in from the keyboard is terminated by pressing Return. The procedure therefore contains two parameters, the first msg$ to contain the character string, and the second RT% which would be set to 'TRUE' if the string is to be followed by Return (ASCII code 13), or 'FALSE' otherwise. The procedure then automatically adds this to the end of the string if required.

Essentially, the procedure consists of a loop which extracts each character in turn from the string given, and uses the OSBYTE call to put this into the keyboard input buffer. The settings for A% and X% can be set up once and for all at the start of the procedure.

One variation of this procedure which can be quite useful (for the more technically minded maybe), is that if you change the value assigned to X% from '0' to '2', the characters are placed not into the keyboard input buffer, but into the RS423 buffer (that is, they will be sent out of the computer via the serial port, and received by any serial device, possibly a printer, connected to that port.

FLASHING COLOURS

As you may know, colours 8 to 15 are the flashing colours (switching alternately between two colours). If you have read your User Guide

in any detail, you will have found that you can control the rate of flashing with *FX9 (for the first colour of the pair) and *FX10 (for the second colour). For example, *FX9,10 sets the flash rate to one fifth of a second (10 fiftieths), while *FX9,0 disables flashing and forces the first colour on the screen. Similarly, *FX10,0 disables flashing and forces the second colour of the pair on the screen, *FX10,5 sets the flash rate to one tenth of a second (5 fiftieths).

However, using other OSBYTE calls you can also get the operating system to tell you what the current settings of the flash rate are by using OSBYTE 195 for the first colour, and OSBYTE 194 for the second. In each case the corresponding rate (the value you would set with *FX9 or *FX10) is returned in the Y register. Thus:

```
OSBYTE=&FFF4
A%=195
rate1=(USR(OSBYTE) AND &FF00) DIV &100
A%=194
rate2=(USR(OSBYTE) AND &FF00) DIV &100
```

will return the two rates as rate1 and rate2 respectively.

VDU STATUS

Another OSBYTE call (117) enables a program to find out the current VDU status. The eight bits of the value returned in the X register indicate current VDU status as in table 1.

Bit	Status
0	Printer output enable by VDU2
1	Page scrolling disabled with VDU15
2	Page scrolling selected with VDU14
3	Text window set
4	Shadow mode in use
5	VDU5 (text at graphics cursor) set
6	Cursor editing in progress
7	Screen output disabled with VDU21

Table 1

To extract the relevant information we now need two steps, the first to extract the relevant byte from the value returned by this OSBYTE call, and then to examine each bit as required, for example:

```
OSBYTE=&FFF4
A%=117
status=(USR(OSBYTE) AND &FF00) DIV &100
IF status AND &01 PRINT"printer enabled"
IF status AND &02 PRINT"VDU15 set"
IF status AND &04 PRINT"VDU14 set"
IF status AND &08 PRINT"Text window set"
IF status AND &10 PRINT"Shadow mode"
IF status AND &20 PRINT"VDU5 set"
IF status AND &40 PRINT"Cursor editing"
IF status AND &80 PRINT"VDU21 set"
```

The whole byte is assigned to the variable status, and in the above example each bit is tested in turn and a suitable message is displayed if the relevant bit is set (equal to 1).

SETTING SHIFT LOCK OR CAPS LOCK

One of the problems that can arise, particularly with single key input, is that a character may be input in either upper or lower case. A program therefore has to cope with this, either by checking both variants (e.g. checking 'YyNn' when seeking a Yes/No response) or by converting the character to upper case, regardless of the case of the character input (by ANDing the ASCII code with &5F).

However, there is another way. A program can itself control the Shift Lock and Caps Lock status of the keyboard using OSBYTE 202. With this call a program can either read the existing status or set a new status. In either case a status byte is needed in which each bit has a particular meaning. However, only two of these matter to us here:

Bit	Status
4	Caps Lock disengaged
5	Shift Lock disengaged

For some reason the settings of these two bits are the reverse of what you might expect:

Bit 5	Bit 4	Meaning
0	0	Caps and Shift Lock
0	1	Shift Lock on
1	0	Caps Lock on
1	1	No Lock

Thus to set Caps Lock on (and Shift Lock off) we need a status value of &20 (bits 5 and 4 making the '2' and unused bits 3 to 0 making the '0'). To set this, proceed as follows:

```
OSBYTE=&FFF4
A%=202:X%=&20
CALL OSBYTE
A%=118
CALL OSBYTE
```

The reason for the two OSBYTE calls is that setting Caps Lock on affects the internal status of the keyboard. On some systems this does not in

itself change the lights at the front of the keyboard to reflect the new state. That's what the second OSBYTE call does (OSBYTE 118). Try it and see - if you don't need OSBYTE 118 then just leave it out. If you need to set Shift Lock on the status value (assigned to X%) is &10, to set both on it's &00, and to set both off it's &30.

In fact, this can also be expressed in *FX calls only by putting:

```
*FX202,<status>
*FX118
```

where <status> is the status value as shown above. This is of course much simpler than the CALL method, but either way the same result is obtained, and a neat one it is too in this case.

USING OSWORD CALLS

You may be forgiven for thinking by this stage that OSBYTE is the only operating system call, but that is not so. In part one (last month) I described the OSCLI routine as my first example (though this is now also a Basic keyword). Another group of operating system routines goes by the name of OSWORD. This routine, like OSBYTE, has a number of variants, the particular one in use again being determined by the value assigned to the A register (A% in Basic). Unlike OSBYTE, there is nothing equivalent to the *FX method of accessing some of those calls.

In general, OSWORD calls are more difficult to handle than OSBYTE because more data is involved, either to be passed to the routine, or returned by the routine. This is normally dealt with by reserving an area of memory for this purpose, and then passing the address of this area. A memory address in the BBC micro always consists of 16 bits which therefore has to be split up and sent as two bytes (we did this with the OSCLI routine last month) usually in the X and Y registers. We will look at just one example of an OSWORD call, but if you read about OSWORD in your User Guide you will find that some of the calls correspond with Basic keywords (for example, the POINT function is the same as OSWORD 9).

READING THE PALETTE

As you may know, the VDU19 call can be used to assign any of the colours possible on a BBC micro to a specific logical colour. For example, mode 4 allows only two colours, which by default are black and white. We could change these to blue and yellow by writing:

```
VDU19,0,4,0,0,0
VDU19,1,3,0,0,0
```

OSWORD 11 allows us to find out what colour has been assigned, for example:

```
DIM block 40
OSWORD=&FFF1
A%=11
X%=block MOD 256
Y%=block DIV 256
?block=1
CALL OSWORD
colour=block?1
```

The DIM statement at the start reserves 40 bytes (ample for our purposes), and Basic will assign the start address of this area of memory to the variable block. The statement:

```
?block=1
```

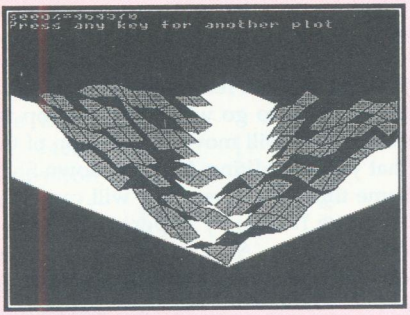
assigns the logical colour we are checking to the first byte of the reserved memory area before OSWORD is called. Afterwards, the second byte of this area contains the value of the associated physical colour which is then assigned to the variable colour. We may seem to be rather making a meal of things, but there is no other way of obtaining this information. And with reflection, the coding is not really that difficult.

Another useful OSWORD call is OSWORD 10 which will read the definition of any ASCII character. In other words, it produces the same eight bytes of the definition that would be needed by VDU23 to define a character. This call is often used by programs which distort or enlarge characters on the screen. On entry, the first byte of the reserved memory area is set to the ASCII code of the character whose definition is required, and on return this byte remains the same, but the following eight bytes contain the definition.

That last example is beginning to go beyond the scope of this article (indeed some may feel we have already exceeded that), but I have tried to show how a knowledge of such operating system routines as OSBYTE and OSWORD can extend what can be achieved in Basic. I hope too, that I have taken you far enough along this particular road for those who are interested to pursue it further themselves.

That concludes our coverage of this topic, at least as far as First Course is concerned. If there is sufficient interest, we can pursue this elsewhere in BEEBUG. Next month we will embark on something new.

The Best of BEEBUG



Basic Booster ROM

SUPER SQUEEZE A program compressor.

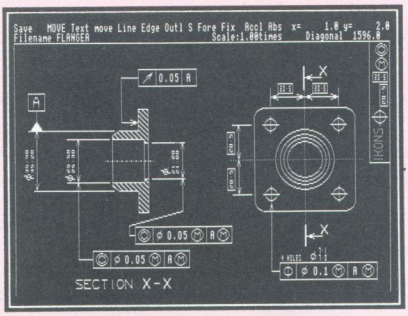
PARTIAL RENUMBER A very useful utility which renumbers a selected block of lines.

PROGRAM LISTER List any program direct from a file.

RESEQUENCER Rearrange the lines in a Basic program - line numbering is automatically adjusted.

SMART RENUMBER Renumber a program so that procedures start at a particular line number.

TEXTLOAD AND TEXTSAVE Save and load a Basic program as text.



Applications II Disc

SHARE INVESTOR
A program which assists decision making when buying and selling stocks and shares.

REAL TIME CLOCK
A real time digital alarm clock displayed for all BBC micros.

RUNNING FOUR TEMPERATURES
A program for calibrating and plotting up to four temperatures.

CROSSWORD EDITOR
Design, edit and solve crosswords with this program.

LABEL PROCESSOR
Design, save and print labels at any size on an Epson compatible printer.

MONTHLY DESK DIARY
A month-to-view calendar which can be used on-screen or printed out.

3D LANDSCAPES
Create computer-generated three dimensional landscapes with this program.

FOREIGN LANGUAGE TESTER
Define foreign characters and test your knowledge of foreign languages.

JULIA SETS
Fascinating displays of Julia sets, the extensions of the Mandelbrot set.

ASTAAD

ENHANCED ASTAAD CAD PROGRAM WITH

- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

Applications I Disc

- BUSINESS GRAPHICS
- VIDEO CATALOGUER
- WORLD BY NIGHT AND DAY
- PHONE BOOK
- PAGE DESIGNER
- PERSONALISED LETTER-HEADS
- MAPPING THE BRITISH ISLES
- SELECTIVE BREEDING
- APPOINTMENTS DIARY
- THE EARTH FROM SPACE
- PERSONALISED ADDRESS BOOK

General Utilities Disc

- PRINTER BUFFER *
- ROM CONTROLLER
- SPRITE EDITOR/ANIMATOR
- MULTI-CHARACTER PRINTER DRIVER FOR VIEW
- MODE 7 SCREEN EDITOR
- MULTI-COLUMN PRINTING
- EPSON CHARACTER DEFINER
- BEEBUG MINIWIMP †
- ROM FILING SYSTEM GENERATOR

*** Master series only. † Requires sideways RAM.**

Please rush me my Best of BEEBUG disc at the members price of £5.75 (non-members price £15)
(ASTAAD disc - members price of £9.95 , non-members price £19.95)

Applications II *Code 1411A (80 track DFS)*	☐	Applications II *Code 1412A (3.5" ADFS)*	☐
Basic Booster ROM *Code 1403A*	☐	Basic Booster Disc *Code 1402A*	☐
ASTAAD Code 1407A *(80 track DFS)*	☐	ASTAAD Code 1408A *(3.5" ADFS)*	☐
General Utilities Disc *Code 1405A (80 track DFS)*	☐	General Utilities Disc *Code 1413A (3.5" ADFS)*	☐
Applications I Disc *Code 1404A (80 track DFS)*	☐	Applications I Disc *Code 1409A (3.5" ADFS)*	☐

Name _____
Address _____
_____
Memership No _____

Price £_____
Postage £ 0.60
Total £_____

I enclose a cheque for £_____ OR please debit my Access, Visa or Connect account, Card No_____/_____/_____/_____ Expiry____/____ Signed_____

Send to: BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Telephone (0727) 40303.

Solids of Revolution

by David Lowndes Williams

INTRODUCTION

A solid of revolution can be created by rotating a profile of a shape about an axis. The program listed here allows you to create and edit such a profile, and to display the shape as a wireframe model or as a solid form with the hidden lines removed.

ENTERING THE PROGRAM

Enter the program and save to disc or tape. The program as listed will work on a standard model B (or Master) with a disc drive. If you want to use it on a tape system the following changes must be made:

DELETE line 2430 (produces file not found message on a disc).

Remove the OSCLI("CAT") from line 180 - this command would catalogue the tape - a process which can be very slow. Line 180 will now read:

```
180 IF G%=56 COLOUR128:CLS:PRINT'"Load
profile:":PROCsave_load("L")
```

USING THE PROGRAM

When you run the program, you will first see the edit page. The screen is divided into two. On the left is the profile and the instructions are on the right. The profile will appear white in colour. The lines either side of the point that you are currently editing are shown in white. The axis of rotation is the dotted line at the centre of the screen.

Pressing 'A' will cause the top to go white and the bottom red. Pressing 'Z' will make the both lines go white. Pressing it again will cause the bottom half to go white and the top red. The cursor keys will move the position of the point that you are editing. Holding down Shift at the same time as a cursor key will cause the point to move faster. By experimenting with the cursor keys, 'A' and 'Z' keys, you should quickly become familiar with their action.

Pressing '1' will display the solid of revolution as a wire frame model. After this press Space to return to the edit page. Alternatively, pressing Escape at any point in the program will return you directly to the edit page. Pressing '2' will display the solid of revolution as a solid form.

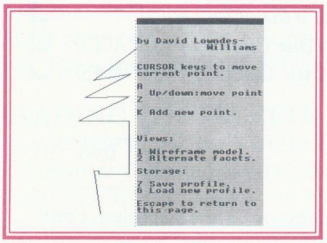
When on the edit page, another point can be added to the profile by pressing 'K'. A line should appear from the last point of the profile (usually at the top) to the middle top of the screen. The cursor keys can then be used to position the point in the desired position. Editing can proceed as usual.

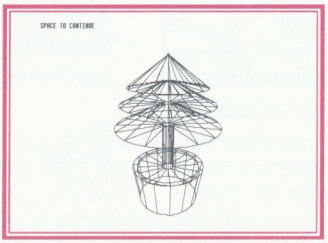
Shift-Escape will allow you to quit from the program. 'S-Escape' will save the screen under the filename "screen". This is to allow you to take a 'snapshot' of the screen at any time (even when actually drawing a solid). In order to recover such a screen type:

```
MODE1
*LOAD"screen"
```

or for the wireframe display:

```
MODE0
*LOAD"screen"
```





LIMITATIONS

The maximum number of points is 20. Any attempt to create more than 20 points on a profile will fail. When using the alternate facets display option, hidden line removal is achieved by starting from the back of the solid and drawing forward. Closer facets are drawn on top of those further behind in the scene. It is possible to enter a profile which will not be reproduced faithfully in this display mode.

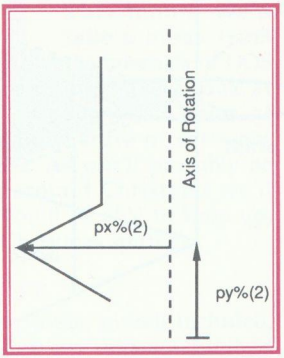


Figure 1

HOW IT WORKS

The edit routine is really editing two arrays, px%(20) and py%(20). All of the information about the shape profile is stored in these arrays.

px%(0)=the number of defined points.
py%(0)=the currently selected point.
px%(1 to 20)=the radius of each point on the profile.
py%(1 to 20)=the height of each point.

First the error trapping and mode are set, then the variables and screen are set up. The program then enters the main loop (lines 130-210). PROCedit is called so that the shape can be edited. This procedure is terminated if a viewing option or a storage option is chosen. These routines are then called from within the main loop. After one of these options has been called, the screen is reset and editing continues.

During execution the Escape key is not disabled, but generates an error message when pressed, which is trapped by the error trapping routine (lines 240-280). This gives the Escape key 'doorbell status' and whenever it is pressed the program is re-directed to the edit routine. The drawing routines are really glorified circle routines. The circles are drawn in 3D space, and transformed onto the screen by the 3D procedures described in a previous article by the author in BEEBUG Vol.6 No.9 p16.

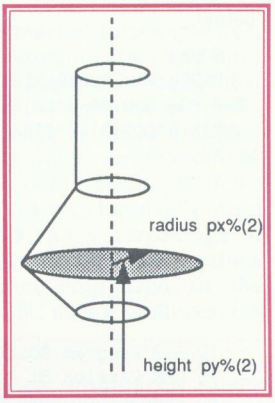
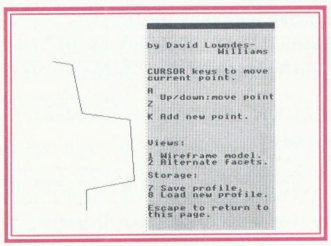


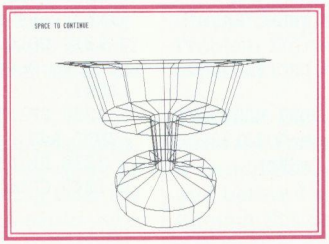
Figure 2

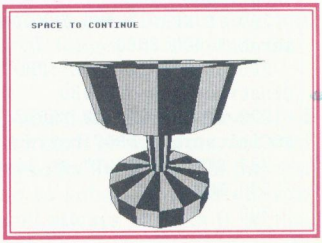
Figure 1 shows a profile, as stored in the arrays. Figure 2 shows the second circle up from the bottom. Figure 3 shows this circle in relation to the axes. This circle is described by:

x=px%(2)*COS(A)
y=px%(2)*SIN(A)
z=py%(2)

where A varies between 0 and 2*PI. These equations describe a circle of radius px%(2), centre (0,0,py%(2)), lying in the plane z=py%(2). This is shown in Figure 3.







The program allows you to create and edit these shapes without any knowledge of the maths, so don't worry if you don't follow this completely. The circle routines are similar to the one line circle routine shown below:

```
MODE1
R=200:VDU29,640;512;:MOVE
R,0:FOR A=0 TO 2*PI STEP 0.
1:DRAW R*COS(A),R*SIN(A) :NE
XT A
```

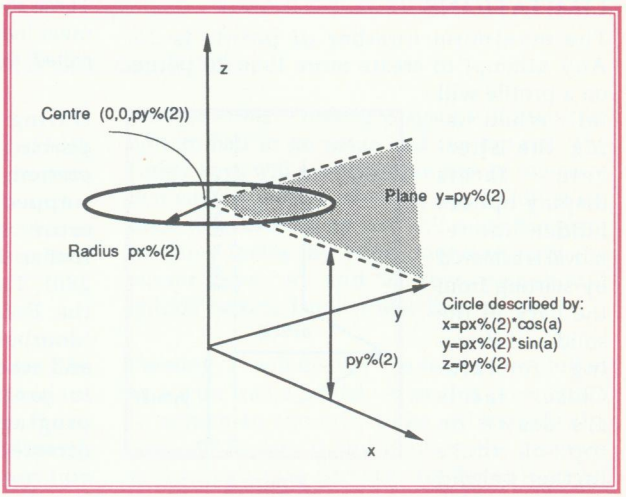


Figure 3 →

```
  10 REM Program SOLIDS
  20 REM Version B1.1
  30 REM Author  David Lowndes Williams
  40 REM BEEBUG  December 1989
  50 REM Program subject to copyright
  60 :
 100 ON ERROR GOTO240
 110 MODE1
 120 PROCinit
 130 REPEAT
 140 PROCedit
 150 IF g%=49 MODE0:PROCwireframe
 160 IF g%=50 MODE1:PROCfacet
 170 IF g%=55 VDU7:COLOUR128:CLS:PRINT"
Save profile:":PROCsave_load("S")
 180 IF g%=56 COLOUR128:CLS:VDU14:OSCLI
("CAT"):VDU15:PRINT'"Load profile:":PROC
save_load("L")
 190 MODE1:PROCinitscreen
 200 PROCinstructions
 210 UNTIL FALSE
 220 END
 230 :
 240 IF ERR=17 AND INKEY-1 OSCLI("FX4")
:MODE7:END
 250 IF ERR=17 AND INKEY-82 THEN *SAVE"
screen"3000 8000
 260 IF ERR=17 MODE1:PROCinitscreen:PRO
Cinstructions:GOTO130
 270 IF ERR=222 OR ERR=204 VDU7:MODE1:P
ROCinitscreen:PROCinstructions:GOTO130
 280 REPORT:PRINT" at line ";ERL
 290 END
 300 :
```

```
1000 DEF PROCinit
1010 DIM px%(20),py%(20)
1020 px%(0)=3:py%(0)=1:px%(1)=100
1030 py%(1)=100:px%(2)=300:py%(2)=200
1040 px%(3)=10:py%(3)=900
1050 PROCinitscreen:PROCinstructions
1060 ENDPROC
1070 :
1080 DEF PROCinitscreen
1090 LOCAL H%:COLOUR128:CLS
1100 FOR H%=0 TO 1024 STEP 16:PLOT69,60
0,H%:NEXT H%
1110 VDU28,19,31,38,0
1120 GCOL0,1:MOVE600-px%(1),py%(1)
1130 FOR a%=2 TO px%(0):DRAW600-px%(a%)
,py%(a%):NEXT
1140 ENDPROC
1150 :
1160 DEF PROCinstructions
1170 COLOUR130:VDU19,2,4,0,0,0
1180 CLS:COLOUR129:PRINT"SOLIDS OF REVO
LUTION":COLOUR130:COLOUR3
1190 PRINT:PRINT"by David Lowndes-"SPC1
4"Williams"'
1200 COLOUR1:PRINT'"CURSOR";:COLOUR3:PR
INT" keys to move current point."'
1210 COLOUR1:PRINT"A":COLOUR3:PRINT"  U
p/down:move point";:COLOUR1:PRINT"Z":COL
OUR3
1220 PRINT:COLOUR1:PRINT"K";:COLOUR3:PR
INT" Add new point."
1230 PRINT'''"Views:"'
1240 COLOUR1:PRINT"1";:COLOUR3:PRINT" W
```

Continued on page 55

512 Forum

by Robin Burton

This month we'll take a break from the internals of DOS and take a look at the good news of the recently improved picture of software availability for the 512. As you'll probably be reading this Forum around Christmas we'll take things easy so you'll be able to keep up, even if you're full of Christmas spirit(s).

THE FACTS OF LIFE

I suppose numerous people, myself included, were at one time hoping that established software producers might eventually decide to take up the 512 cause. However, as time passed it became obvious this just wasn't going to happen.

The unfortunate reality is that the 512 market simply isn't big enough to attract the interest of large Acorn software suppliers, who see better returns and less risk in other areas. In addition they don't have the necessary expertise and acquiring it would be expensive.

This situation exists partly because Acorn didn't exploit the machine's potential as they should have years ago, and equally because there's no shortage of PC software anyway, though it doesn't always solve 512 problems. Sadly, we are the beneficiaries of this short-sighted policy, and as I've said in the past, apart from the notable exceptions of Dabs Press and BEEBUG, we've been on our own.

As you may have read in Vol.8 No.5, I'm now trying to do my bit (see review of Essential Software's Ramdisc Utility), but I'm delighted to be able to say that there are also some other recent software releases for the 512, and other things are planned by at least one of the parties. Perhaps with a bit of encouragement the 512 will refuse to roll over and die after all as we enter the '90s.

These new releases are all from small operations, simply because in the first place they don't or didn't have to make their entire living from the software. Effectively they are enthusiasts who have written software for the 512 mainly because they wanted to, rather than being driven purely by commercial considerations.

To help put the 512 market in perspective, remember that 512 Forum offers the only continuous and regular support for the 512 that has ever appeared in the entire Acorn press (and you can check back as far as you like, there's never been any other). In spite of this, my offer of a free copy of 512 BBCBASIC in the June issue (Vol.8 No.2) has produced only about 275 requests to date. As a percentage of BEEBUG's current membership that's quite a healthy figure when compared to the percentage of 512s to BBC micros produced (but where are the rest?).

The trouble is though, compared to the estimated 1.7 million 6502 based BBC micros, 275 is insignificant (even though it didn't seem like it when I was copying your discs!), and that's why software suppliers haven't been interested in the 512.

As I said, the picture has now changed for the better, and with luck and user support this might continue. Next though, I must give my excuses about why some things take longer to appear in the Forum than might be expected. This is thinly disguised as an overview of how the Forum reaches you.

EXCUSES

One of the earliest recipients of a copy of 512 BBCBASIC was Lawson Wakefield, who mentioned that he produces a PC package and that he'd also produced a version for the 512. I wrote to Lawson for more information and it's included below at last. I must apologise to him for the delay since his letter in late July, but Problem Solver took longer to deal with than expected.

The first inescapable fact is that generally I write the Forum a month or so before you read it. In addition, I try to fix the content of the next one before I need to submit the current one (though this doesn't always work out), so that

at least a thread of continuity can be maintained. As an illustration of the delay, the last Forum was written in mid-September, but you may well be reading this one after having your Christmas dinner.

The unfortunate result of all this is that typically an item may not appear for three or four months after I first hear about it. It's also why most magazines carry a news section, space reserved to include late, up to date items that needn't be limited by standard production schedules.

That's enough of excuses, back to software.

INTERACTER

Fortran, which stands for 'FORmula TRANslation' is a programming language designed to be especially efficient for scientific and mathematical applications, as the name suggests. Although never the most popular of languages because of its specialist nature, it originally appeared on mainframe computers in the sixties and successfully survived and grew to the present day. As micros have evolved and become more powerful, so Fortran has become available on more and more machines, and the PC (hence the 512) is amongst them. Of course, in common with all languages Fortran has its areas of weaknesses. Let me explain.

For example, the Basic we know is designed primarily for easy string manipulation and so console I/O is also very easy too, but complex calculations can become very heavy and tedious, both to encode and during execution, while random access disc filing is non-existent: it's entirely a 'do it yourself' job. Since console I/O is included in Basic, any version used must generally have peculiarities built in, in order to cater for the hardware employed, in other words it's machine specific. You'll have been reminded of this if you've tried to convert programs between, say, BBC Basic and Microsoft Basic or similar on the 512.

Fortran, on the other hand, is designed specially for complex calculation and source code portability, but like other languages designed with source portability in mind (perhaps 'C' is the most well known) user I/O cannot be an integral part of the language specification.

This aspect of each implementation is therefore left to each supplier to deal with. In respect of 'C', for example, standard library subroutines (like 'STDIO' which stands for standard I/O) will be included in the package by the supplier to cater for the vagaries of the hardware employed. These library routines are called from user code to perform I/O tasks, to save you from having to write hardware handling code from 'scratch' for yourself.

Fortran is similarly conceived but, probably due to its more specialised application, the input-output facilities included as standard are truly primitive. Any sort of presentable or 'user friendly' screen interface like a menu, or formatted output to a printer, becomes really hard work. Worse, having written such facilities for your own machine they won't help you on any other. Change machines and you must start again.

To the rescue comes *Interacter* from Interactive Software Services. This is a software development package designed to provide a wide range of ready-written console and I/O subroutines to do most of the hard work for you. In addition, to aid source portability, compatible versions of *Interacter* are available for a range of mini computers driving terminals and workstations (e.g. Hewlett Packard, SUN, DEC VAX) plus a variety of IBM PCs covering the various screen display types, MDA, CGA, EGA, MCGA and VGA. The most interesting point for us though, is that the range of machines includes the 512 (using either a model B or a Master host) and the Archimedes.

Versions of the Fortran compiler supported by *Interacter* for PCs and the 512 are Prospero and Microsoft v4.1, while for the Archimedes it's Acornsoft's release 2, either RISC OS, or Arthur if anyone still uses it.

For business or educational use, the licence terms for *Interacter* look sensible and programs produced with its help don't incur royalties or additional fees, something to beware of with some development packages. *Interacter* facilities and prices vary with your use, for example, on a 512 the 'personal' version is what you need at £99.00 plus VAT. At the other end of the scale, if your employer runs a multi-user (i.e. 50 plus) mini, the price grows to £4000 which is serious

money, but is actually slightly cheaper on a per screen basis.

I'm not a Fortran user, but I am familiar with it because of my mainframe activities. I've run the 512 demos and I must say my conclusion is that this development package looks extremely interesting, comprehensive and professionally produced. If you or your employer/school/college write scientific applications using Fortran, contact Lawson Wakefield at I.S.S. for an information pack, demonstration discs and full price details.

MOUSE DRIVER

Another BEEBUG member (and 512 BBCBASIC recipient) Cliff Bowman, has started a company called Tull Computer Services, which you may have seen advertised recently.

T.C.S.'s first product is an MS compatible mouse driver for the 512, and I've been having a look at it. To a certain extent there's not much to say in a report on a mouse driver; either it does the job or it doesn't, and Tull's program does.

However, as is so often the case with the 512, things aren't so black and white, though this is definitely not the fault of the mouse driver. By definition a mouse driver is a support program for a main application, and to work correctly the main application must interface with a mouse legally.

Just as some packages use peripheral addressing techniques which cause them to fail totally in the 512, so it appears that some may try to read the mouse hardware directly, instead of using INT 33h, the legal mouse interrupt. There's no good reason for this, INT 33h offers sufficient facilities to configure a mouse in virtually any way you might wish. Nevertheless, some programs insist on doing their own thing and the result is that these won't work with a mouse on the 512.

I point this out only so that no-one thinks that because a mouse driver is properly written it guarantees everything will work with everything else in the 512 - it just isn't true. However, if you do know that a package works in the 512, and you also know it works in a PC with an MS (standard) mouse driver you should be on a pretty safe bet.

Tull are quite open about this fact in their manual, and include a list of programs which have been tested successfully, as well as a couple (one really) that haven't, so I've included the current list here for you. The manual also includes comprehensive details of INT 33h functions for those who want to write their own programs to interface with the mouse.

Compatible PC packages include:

Microsoft Word v4.0
Norton Editor v1.3C
Autoroute by Nextbase
Autosketch by Autodesk
Gem 3.0 and 3.11 by Digital Research
Deluxe Paint II by Electronic Arts
Elite by Firebird

Of these, one or two are memory hungry and ideally require an expanded 512, but that's nothing to do with the mouse driver. As mentioned previously, Elite also requires Problem Solver, again not the mouse driver's fault. Also note that in some versions of Autoroute the map doesn't work at all in the 512, while in others it's fine.

Incompatible PC packages include:

The 512's packaged Gem (no surprise)
Tubocad 1.5 by Pink Software

I expect that the list of tested software will grow as the numbers of users increase, especially if they keep T.C.S. informed. If your package isn't listed check with T.C.S. as they may have more information by the time you read this. Cliff would prefer written enquiries, but if you must phone, do so in the evenings. The T.C.S. mouse driver costs £30.00 complete.

ADDRESSES

INTERACTER - I.S.S. 25, St. Michael's Close, Penkridge, Stafford ST19 5AD.
Tel. (0785) 715588

MOUSE DRIVER - T.C.S., 115 Gammons Lane, North Watford, Herts WD2 5JD.
Tel. (0923) 662240

Postscript Vector Graphics

This month, Willem van Schaik shows how BBC Basic's MOVE, PLOT and DRAW functions can be converted into PostScript equivalents for output to a laser printer.

INTRODUCTION

As stated in the previous article (BEEBUG Vol.8 No.6), PostScript can generate images using vectorised definitions, as well as the bitmaps described there. In fact, this is the most common way of using a page description language like PostScript, while bitmaps are used more with equipment like scanners or video-cameras.

Vectorised drawings sounds very complicated, but it simply means specifying a picture with co-ordinates, much as you do in BBC Basic with commands like PLOT, DRAW or MOVE. Not only graphics, but text can also be handled, very much as in BBC Basic with the VDU 5 command by placing text at the graphics cursor.

In this article, I will describe a program which converts a program in BBC Basic using MOVE, DRAW and PLOT into an equivalent PostScript program for driving a laser printer. There will also be a test program to check your conversion before producing any printout, and a demo program to show how it all works.

CO-ORDINATE SYSTEMS

First some theory: one consequence of PostScript's flexibility is the facility to create your own co-ordinate system. The default system is based on a unit size of 1/72 inch with the origin in the lower left-hand corner of the (upright) page. The unit-size can be altered with the "x-factor y-factor scale" command (note the reverse Polish or postfix notation - see last month's Workshop for more details), while the origin can be changed with the "x y translate" command. With the "degrees rotate" command we can, for example, change an image from *portrait* to *landscape* format.

This can be illustrated with a short PostScript program which could be used to create a co-ordinate system for plotting a function like y=sin(x). Suppose you want a landscape shaped picture with the origin in the middle of the page, and with a unit-size of 1 centimetre:

```
Instruction          Comment
28.3 28.3 scale      % 1 cm = 28.3 * 1/72 inch
                     % Scaling in x-direction and y-direction are here
                     % equal but they can be different
10.4 14.8 translate  % origin in centre of A4 page
                     % x and y must be given in centimetres, because
                     % translate comes after the scale command
90 rotate            % in degrees counter-clockwise
```

When we have created the co-ordinate system most suitable for our application, we can make temporary changes to it by a *save and restore* mechanism. The PostScript commands for this are respectively *gsave* and *grestore*. With *gsave*, the current co-ordinate system is put on the stack, while with *grestore*, the current co-ordinate system is replaced by the one taken from the stack. This feature of PostScript is used a lot in the procedure-library that follows later.

A PATH

Drawing an object in PostScript is not as straightforward as the PLOT commands in BBC Basic. This is caused by the need for flexibility. In the process of creating an object - for example a box - three steps can be distinguished.

First, a number of parameters can be set that will determine the appearance of our object. For lines you can think of the thickness and the colour (or grey-value) of the line, or perhaps you want to make a choice between a dotted or a dashed line, etc. For filled objects, you can define the colour or greyness of the object.

Second, you must program the shape of the object in the current co-ordinate system. This is done using a so called *path*. This describes the surroundings of the object to be defined without specifying its appearance. The definition of a path starts with a *newpath* command, followed by one or more absolute or relative DRAW commands, which in PostScript are called respectively *lineto* and *rlineto*. MOVE commands can similarly be given with *moveto* and *rmoveto*. If a closed shape is to be defined, the last command should be *closepath*.

Even after the path has been specified, no graphic object has yet been generated, because its appearance must first be specified with *stroke* for an outline drawing, or *fill* for a solid object. These two commands really create the object. When this happens the parameters of step one become valid and define, for example, the colour of the object or the thickness of its outline.

TEXT

One of the specialities of PostScript is the use of outline fonts instead of bitmapped fonts. This means that for any font, only one definition is

needed, independent of the size or slope of the text that is to be printed using that font. Each time these vectorised outlines are used, they are transformed into bitmaps, a process which requires the kind of high performance microcomputer (mostly Motorola 68000 based) incorporated in PostScript laser printers.

All but one of the fonts used by PostScript are proportionally spaced. This means that the width of an 'i' and a 'm' are different, which improves the readability of the text. However, a BBC screen uses characters that are evenly spaced. Therefore, in what follows, I have used Courier as the font as this is the only non-proportionally spaced one. To get the best comparable appearance I have chosen the bold version of Courier.

USING BBCPOST

The first program listed here (Listing 1) is in the form of a procedure library, called BBCPOST, which consists of 12 procedures which are meant to be added to your own BBC Basic programs. Type in and save these routines as BBCPOST.

For each BBC Basic graphics command, there is a comparable procedure that, instead of writing to the screen, generate some corresponding lines of PostScript. This results, later on, in the same picture on paper as would have appeared on screen. Just like the PostScript screen dump program in BEEBUG last month, this PostScript program writes to screen and disc using a *SPOOL command.

Let's assume that you have an existing program that creates a graphics display. To make the use of BBCPOST possible, your program should use only the following Basic-commands:

```
MODE n, CLG, GCOL 0,colour,
VDU 24,lx;ly;rx;ry;, VDU29,x;y;,
MOVE x,y, DRAW x,y, PLOT 69,x,y,
PLOT 85,x,y and PRINT string$.
```

What you must do is replace all these commands by their corresponding procedure calls. So replace VDU 29,640;256; by PROCPSvdu29(640,256) or DRAW 0,400 by PROCPSdraw(0,400). If you spool your Basic program to disc, and then use a word processor or editor for the changes, it involves much less work than it sounds.

Before you call any of these procedures in your program, you must first call the procedure, PROCPSinit, specifying the name of the SPOOL file to use and the width of the picture on paper. You must also terminate your program with PROCPSexit. The logical value passed to this procedure determines if the file will be a normal or an encapsulated PostScript file (see last month).

Normally, the use of the MODE command near the start of a program is optional, but the use of the procedures PROCPSmode(n) and PROCPSinit() at the start is obligatory. Some scaling and initialisation concerning the width of texts is accomplished in this latter routine.

A few other remarks on the use of these procedures may be appropriate here. To save printer toner, it is wise to have (also on screen) a picture with black objects on a white background. Therefore, if possible, start your program with:

```
GCOL 0,135 : GCOL 0,0 : CLG
```

or with the corresponding calls in BBCPOST, namely:

```
PROCPSgcol(0,135) : PROCPSgcol(0,0) :
PROCPSclg
```

When you use PROCPSprint(string$), keep in mind that the position of the text on the paper is based on the VDU 5 principle in BBC Basic. Therefore, always call PROCPSmove(x,y) first before you call PROCPSprint.

TESTING

The process of creating the PostScript spoolfile, then copying it to a computer to which a PostScript printer is connected, and then printing it, can be a long one. The option of testing the PROCPS... procedure calls beforehand is therefore very handy. A set of equivalent routines, called TSTPOST, that generate the corresponding BBC Basic commands is provided as Listing 2. Enter and save these routines with the name TSTPOST. Thus using the TSTPOST library routine PROCPSplot85(x,y) will draw a triangle using a PLOT 85,x,y command.

To make everything work, the line numbers of both procedure libraries start at 10010, and your Basic programs should use line number ranges below this. The steps involved are then as follows:

First, write your program, incorporating the right procedure calls. Then type:

```
PRINT ~TOP-2    (resulting in XXXX)
*LOAD TSTPOST XXXX
```

RUN the program and examine the picture generated on screen. When you are satisfied with the result type:

```
DELETE 10000,11000
*LOAD BBCPOST XXXX
```

Now RUN the program and the PostScript program will be created on disc.

In the above paragraphs, reference has often been made to "your program". As an example, a program called PSCOLOUR is included with this article (Listing 3). It demonstrates the possible quality of PostScript printouts using BBCPOST. Both the graphics and the text facilities of the procedure library are used.

Using TSTPOST there is one small inconvenience. A BBC Basic MODE command cannot be given from inside a procedure, but must be issued in the main program. Therefore, after adding TSTPOST, you must temporarily change the call PROCPSmode(n) to MODE n. If you don't, you get a warning and the program stops execution. After testing don't forget to replace MODE n by PROCPSmode(n) or else the program will not work properly.

I have implemented what I consider to be the 10 most used graphics commands. Given the framework provided, it would not be too difficult to add further procedures dealing with other commands. If that is too daunting, you will often be able to achieve the same result with a suitable combination of existing functions.

Next month we shall conclude this series by presenting a program for generating PostScript files from mode 7 screens.

Listing 1

```
10010 REM Program BBCPOST
10020 REM Version B 2.0
10030 REM Author  Willem van Schaik
10040 REM BEEBUG  December 1989
10050 REM Program subject to copyright
10060 :
10070 DEF PROCPSinit(PSfile$,PSwidth)
10080 @%=&000010
10090 OSCLI("SPOOL "+PSfile$)
10100 PRINT"%!PS-Adobe-1.0"
10110 PRINT"%%DocumentFonts: Courier-Bol
d"
10120 PRINT"%%Title: BBC-Basic picture -
 ";PSfile$
10130 PRINT"%%Creator: Willem van Schaik
"
10140 PRINT"%%Pages: 1"
10150 PRINT"%%BoundingBox: ";
10160 llx%=302+(28.8*PSwidth/25.4)
10170 lly%=418-(36*PSwidth/25.4)
10180 urx%=298-(28.8*PSwidth/25.4)
10190 ury%=422+(36*PSwidth/25.4)
10200 PRINT ;llx%;" ";lly%;" ";urx%;" ";
ury%
10210 PRINT"%%EndComments"
10220 @%=&020110
10230 PRINT"/cp {closepath} def"
10240 PRINT"/f0 {/Courier-Bold findfont
[26 0 0 65 0 0] makefont setfont} def"
10250 PRINT"/f1 {/Courier-Bold findfont
[52 0 0 65 0 0] makefont setfont} def"
10260 PRINT"/f2 {/Courier-Bold findfont
[104 0 0 65 0 0] makefont setfont} def"
10270 PRINT"/fi {fill} def"
10280 PRINT"/lt {lineto} def"
10290 PRINT"/mt {moveto} def"
10300 PRINT"/np {newpath} def"
10310 PRINT"/p0 {2 0 rl 0 4 rl -2 0 rl c
p fi} def"
10320 PRINT"/p1 {4 0 rl 0 4 rl -4 0 rl c
p fi} def"
10330 PRINT"/p2 {8 0 rl 0 4 rl -8 0 rl c
p fi} def"
10340 PRINT"/rl {rlineto} def"
10350 PRINT"/rm {rmoveto} def"
10360 PRINT"/rs {grestore gsave} def"
10370 PRINT"/sc {setrgbcolor} def"
10380 PRINT"/st {stroke} def"
10390 PRINT"/tr {translate} def"
10400 PRINT"%%EndProlog"
10410 PRINT"300 420 translate"
10420 PRINT"90 rotate"
10430 PRINT"72 25.4 div dup scale"
10440 PRINT ;PSwidth;" 2 div neg dup 1.2
5 div translate"
10450 PRINT ;PSwidth;" 1280 div dup scal
e"
10460 PRINT"np 0 0 mt 1279 0 lt 1279 102
3 lt 0 1023 lt cp clip"
10470 PRINT"4 setlinewidth"
10480 PRINT"gsave"
10490 PRINT"0 0 0 sc"
10500 PRINT"np 0 0 mt 1279 0 lt 1279 102
3 lt 0 1023 lt cp fi"
10510 PRINT"1 1 1 sc"
10520 PSlastx=0:PSlasty=0
10530 PScurrentx=0:PScurrenty=0
10540 PSlx=0:PSly=0
10550 PSrx=1279:PSry=1023
10560 PSrgb$="1 1 1":PSbackrgb$="0 0 0"
10570 ENDPROC
10580 :
10590 DEF PROCPSmode(PSm)
10600 PSmode=PSm
10610 PRINT"0 0 0 sc"
10620 PRINT"np 0 0 mt 1279 0 lt 1279 102
3 lt 0 1023 lt cp fi"
10630 PRINT"1 1 1 sc"
10640 PScurrentx=0:PScurrenty=0
10650 PSlx=0:PSly=0
10660 PSrx=1279:PSry=1023
10670 ENDPROC
10680 :
10690 DEF PROCPSclg
10700 PRINT PSbackrgb$;" sc"
10710 PRINT"np ";PSlx;" ";PSly;" mt ";PS
```

```
rx;" ";PSly;" lt ";
10720 PRINT ;PSrx;" ";PSry;" lt ";PSlx;"
 ";PSry;" lt cp fi"
10730 PRINT PSrgb$;" sc"
10740 PScurrentx=0:PScurrenty=0
10750 ENDPROC
10760 :
10770 DEF PROCPSgcol(PSa,PSc)
10780 IF PSc<128 THEN PSforeground=TRUE
ELSE PSforeground=FALSE : PSc=PSc-128
10790 IF PSmode=0 OR PSmode=4 THEN PScol
our=7*PSc
10800 IF PSmode=1 OR PSmode=5 THEN PScol
our=2^PSc-1
10810 IF PSmode=2 THEN PScolour=PSc MOD
8
10820 PSrgb$=STR$(PScolour MOD 2)+" "
10830 PSrgb$=PSrgb$+STR$((PScolour MOD 4
) DIV 2)+" "
10840 PSrgb$=PSrgb$+STR$(PScolour DIV 4)
10850 IF PSforeground THEN PRINT PSrgb$;
" sc" ELSE PSbackrgb$=PSrgb$
10860 ENDPROC
10870 :
10880 DEF PROCPSvdu24(PSlx,PSly,PSrx,PSr
y)
10890 PRINT"rs"
10900 PRINT PSrgb$;" sc"
10910 PRINT"np ";PSlx;" ";PSly;" mt ";PS
rx;" ";PSly;" lt ";
10920 PRINT ;PSrx;" ";PSry;" lt ";PSlx;"
 ";PSry;" lt cp clip"
10930 ENDPROC
10940 :
10950 DEF PROCPSvdu29(PSx,PSy)
10960 PRINT"rs"
10970 PRINT PSrgb$;" sc"
10980 PRINT ;PSx;" ";PSy;" tr"
10990 ENDPROC
11000 :
11010 DEF PROCPSmove(PSx,PSy)
11020 PSlastx=PScurrentx:PSlasty=PScurre
nty
11030 PScurrentx=PSx:PScurrenty=PSy
11040 ENDPROC
11050 :
11060 DEF PROCPSdraw(PSx,PSy)
11070 PSlastx=PScurrentx:PSlasty=PScurre
nty
11080 PScurrentx=PSx:PScurrenty=PSy
11090 PRINT"np ";PSlastx;" ";PSlasty;" m
t ";PScurrentx;" ";PScurrenty;" lt st"
11100 ENDPROC
11110 :
11120 DEF PROCPSplot69(PSx,PSy)
11130 PSlastx=PScurrentx:PSlasty=PScurre
nty
11140 PScurrentx=PSx:PScurrenty=PSy
11150 PRINT"np ";PScurrentx;" ";PScurren
ty;" mt ";
11160 IF PSmode=0 OR PSmode=4 THEN PRINT
"p0"
11170 IF PSmode=1 OR PSmode=5 THEN PRINT
"p1"
11180 IF PSmode=2 THEN PRINT"p2"
11190 ENDPROC
11200 :
11210 DEF PROCPSplot85(PSx,PSy)
11220 PSbeforeX=PSlastx:PSbeforeY=PSlast
y
11230 PSlastx=PScurrentx:PSlasty=PScurre
nty
11240 PScurrentx=PSx:PScurrenty=PSy
11250 PRINT"np ";PSbeforeX;" ";PSbeforeY
;" mt ";PSlastx;" ";PSlasty;" lt ";
11260 PRINT ;PScurrentx;" ";PScurrenty;"
 lt cp fi"
11270 ENDPROC
11280 :
11290 DEF PROCPSprint(PSstr$)
11300 IF PSmode=0 THEN PRINT"f0 ";
11310 IF PSmode=1 OR PSmode=4 THEN PRINT
"f1 ";
11320 IF PSmode=2 OR PSmode=5 THEN PRINT
"f2 ";
11330 PRINT ;PScurrentx;" ";PScurrenty;"
 30 sub mt (";PSstr$;") show"
11340 ENDPROC
11350 :
11360 DEF PROCPSexit(PSpage)
11370 IF PSpage THEN PRINT"showpage"
11380 PRINT"%%Trailer"
11390 PRINT"grestore"
11400 OSCLI("SPOOL")
11410 ENDPROC
11420 :
```

Listing 2

```
10010 REM Program TSTPOST
10020 REM Version B 2.0
10030 REM Author  Willem van Schaik
10040 REM BEEBUG  December 1989
10050 REM Program subject to copyright
10060 :
10070 DEF PROCPSinit(PSfile$,PSwidth)
10080 ENDPROC
10090 :
10100 DEF PROCPSmode(PSm)
10110 PRINT'"A mode-change is not allowe
d inside"
10120 PRINT"a procedure. Therefore, to p
erform"
10130 PRINT"this test, replace in your p
rogram"
10140 PRINT"the statement PROCPSmode(";P
Sm;") by the"
10150 PRINT"statement MODE ";PSm;"."
10160 STOP
10170 ENDPROC
10180 :
10190 DEF PROCPSclg
```

```
10200 CLG
10210 ENDPROC
10220 :
10230 DEF PROCPSgcol(PSa,PSc)
10240 GCOL PSa,PSc
10250 ENDPROC
10260 :
10270 DEF PROCPSvdu24(PSlx,PSly,PSrx,PSr
y)
10280 VDU 24,PSlx;PSly;PSrx;PSry;
10290 ENDPROC
10300 :
10310 DEF PROCPSvdu29(PSx,PSy)
10320 VDU 29,PSx;PSy;
10330 ENDPROC
10340 :
10350 DEF PROCPSmove(PSx,PSy)
10360 MOVE PSx,PSy
10370 ENDPROC
10380 :
10390 DEF PROCPSdraw(PSx,PSy)
10400 DRAW PSx,PSy
10410 ENDPROC
10420 :
10430 DEF PROCPSplot69(PSx,PSy)
10440 PLOT 69,PSx,PSy
10450 ENDPROC
10460 :
10470 DEF PROCPSplot85(PSx,PSy)
10480 PLOT 85,PSx,PSy
10490 ENDPROC
10500 :
10510 DEF PROCPSprint(PSstr$)
10520 VDU 5
10530 PRINT PSstr$
10540 ENDPROC
10550 :
10560 DEF PROCPSexit(PSpage)
10570 ENDPROC
10580 :
```

Listing 3

```
   10 REM Program PSCOLOUR
   20 REM Version B 4.1
   30 REM Author  Willem van Schaik
   40 REM BEEBUG  December 1989
   50 REM Program subject to copyright
   60 :
   70 ON ERROR CLOSE#0:OSCLI("SPOOL"):MO
DE3:REPORT:PRINT" at line ";ERL:END
   80 PROCPSinit("$.PSCOLOR",210)
   90 PROCPSmode(2)
  100 PROCsequence:PROCbackground
  110 FOR i%=0 TO 7:PROCblock(col%(i%),9
6*i%+504,96*i%+568,0,1024):NEXT i%
  120 FOR i%=0 TO 7:PROCblock(col%(7-i%)
,0,1280,112*i%+16,112*i%+88):NEXT i%
  130 PROCtext:PROCtitle
  140 PROCPSexit(TRUE)
  150 END
  160 :
  170 DEF PROCsequence : LOCAL i%
  180 DIM col%(7)
  190 FOR i%=0 TO 6 STEP 2
  200 col%(i%)=i%/2:col%(i%+1)=i%/2+4
  210 NEXT i%
  220 ENDPROC
  230 :
  240 DEF PROCbackground
  250 PROCPSgcol(0,135):PROCPSclg
  260 PROCPSgcol(0,0)
  270 PROCPSmove(0,0)
  280 PROCPSmove(1279,1023)
  290 PROCPSplot85(1279,0)
  300 ENDPROC
  310 :
  320 DEF PROCblock(c%,x1%,x2%,y1%,y2%)
  330 PROCPSgcol(0,c%)
  340 PROCPSmove(x1%,y1%)
  350 PROCPSmove(x2%,y1%)
  360 PROCPSplot85(x1%,y2%)
  370 PROCPSplot85(x2%,y2%)
  380 ENDPROC
  390 :
  400 DEF PROCtext : LOCAL i%,j%,k%
  410 PROCtextarray
  420 FOR i%=0 TO 7
  430 j%=7-i%:k%=i%+5
  440 IF k%>7 THEN k%=k%-8
  450 PROCword(col%(k%),24,112*j%+64,col
$(col%(i%),0))
  460 PROCword(col%(k%),96*i%+504,1004,S
TR$(col%(i%)))
  470 PROCword(col%(k%),96*i%+504,936,co
l$(col%(i%),1))
  480 NEXT i%
  490 ENDPROC
  500 :
  510 DEF PROCword(c%,x%,y%,tx$)
  520 PROCPSgcol(0,c%)
  530 PROCPSmove(x%,y%):PROCPSprint(tx$)
  540 ENDPROC
  550 :
  560 DEF PROCtextarray : LOCAL i%,j%
  570 DIM col$(7,1)
  580 FOR j%=0 TO 1:FOR i%=0 TO 7
  590 READ col$(i%,j%)
  600 NEXT i%:NEXT j%
  610 DATA Black,Red,Green,Yellow,Blue,M
agenta,Cyan,White
  620 DATA B,R,G,Y,B,M,C,W
  630 ENDPROC
  640 :
  650 DEF PROCtitle
  660 PROCblock(2,32,464,892,1004)
  670 PROCword(12,56,984,"P s - ")
  680 PROCword(11,56,984," o t  ")
  690 PROCword(11,56,936,"S r p ")
  700 PROCword(12,56,936," c i t")
  710 ENDPROC
  720 :
```

RISC USER

The Archimedes Magazine & Support Group

Risc User is enjoying the largest circulation of any magazine devoted solely to the Archimedes range of computers. Now in its third year of publication, it provides support to schools, colleges, universities, industry, government establishments and private individuals. Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines. A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer, particularly at this time while documentation on the Archimedes and RISC OS is still limited.

Here are just some of the topics covered in more recent issues of RISC User:

DESKTOP CALCULATOR
A calculator which works in decimal, binary or hex.

PRINTERS, DRIVERS AND FONTS
A look at the major elements involved in getting a smart printout from your Archimedes.

THE ARM 3 RISC PROCESSOR
An in-depth look by the designer of this exciting development.

ACORN'S ANSI C
A look at the latest release of Acorn's C compiler.

SCREEN COMPRESSOR
A module for the speedy and efficient compression and expansion of screen displays.

FLOATING POINT INSTRUCTION SET
An introduction to using floating point arithmetic in ARM assembler.

MASTERING THE WIMP
A major series for beginners to the WIMP programming environment.

RISC USER DESKTOP DIARY
A multi-tasking Desktop diary facility.

UNDER THE LID
A major series explaining the hardware that makes up the Archimedes.

DESKTOP HOTKEYS UTILITY
An application providing single key shortcuts from the Desktop.

MINERVA'S MULTISTORE
A review of the latest package from System Delta.

DESKTOP BASIC HANDLER
A Desktop utility to convert basic programs into their text equivalents and vice versa.

ARCADE
A round-up of the latest games for the Arc.

INTRODUCING LASER PRINTERS
A look at the workings of laser printers and Page Description Languages.

Don't delay - Phone your instructions now on (0727) 40303

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.10 (overseas see below).

Name:.....
Memb No:.....
Address: .....
.....
.....

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.10
Rest of Europe and Eire	£12.00
Middle East	£14.00
Americas and Africa	£15.00
Elsewhere	£17.00

I wish to receive both BEEBUG and RISC User. *I enclose a cheque for £* ..... *or alternatively:*

I authorise you to debit myACCESS/Visa/Connect account: /_____/ _____/ _____/ _____/

Signed: ..... *Card Expiry Date:* /_____/ _____/

Send to: RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, or telephone (0727) 40303, or FAX (0727) 60263

Writing a Compiler (Part 2)

David Spencer continues his discussion of how you can write a compiler.

This month we move on to consider the *Syntax Analyser* phase of our simple compiler. As outlined last time, the purpose of this section of the compiler is to pick out chunks of the source program that form a valid part of the language being compiled. The example given last month was that of a FOR statement in Basic, where as soon as the syntax analyser has found the token for a 'FOR', it knows what components must follow for this to be a genuine FOR statement. It can then proceed to pick these out, and check that they do indeed match expectations.

Much of the material this month may be a bit hard-going, but by reading it a few times, following the examples, and trying a few of your own, it should be relatively straightforward to follow.

GRAMMARS

It is obvious that in order to pick out valid constructs for a particular source language, we need some way of specifying the syntax of all the possible constructs allowed by the language. For example, Basic can only check and decode FOR statements because it knows what one should look like. It is equally obvious that whatever method we choose to represent the language, it must be accurate and precise, otherwise the entire compiler will become unwieldy. Additionally, it must be relatively easy to verify that the chosen system does indeed represent all the constructs of the language, and only those constructs. A final criterion is that the chosen notation must allow the syntax analyser to describe the structure of the source program in enough detail to convert it to machine code (or whatever the target language is).

To meet all the above criteria, programming languages are usually formally specified using a notation known as *Context Free Grammars*, or just *Grammars* for short (also called Bakus-Naur Form). You are in fact, already familiar with one such grammar - that used to describe the English language, although I doubt many people could write down the rules of English grammar in any concise and precise form. (The term Context free arises from the fact that the grammar specifies a particular construct of the language independent of its surroundings. Compare this with the lexical analyser from last month, in which the interpretations of a '-' as negation or subtraction depended on its position in the input string.)

Rather than explain abstract theory, we will start by presenting the grammar that represents the language we are going to compile - simple expressions. This is written as:

expr := *expr* + *term* | *expr* - *term* | *term*
term := *term* * *factor* | *term* / *factor* | *factor*
factor := **number** | **(** *expr* **)** | - *factor*

This might look daunting, but is in fact quite easy to follow. Each line states how a particular language construct, called a non-terminal, is made up of other constructs. Such a definition is called a production. The name on the left hand side of the := is the name of the non-terminal, while the right hand side specifies the possible ways in which that non-terminal can be made up. When there are a number of different options, these are separated by a '|' which means 'or'. The ':=' is read as 'derives', or 'is made up of'. On the right hand side of the production, names in italics represent other non-terminals, just like those on the left hand side, while items in bold correspond to actual input symbols, as returned by the lexical analyser.

As an example, the first line says that an expression (*expr*) can be made up of another expression, followed by a '+', followed by a non-terminal named *term*, or an expression, a '-' and a *term*, or just a *term*. The second line defines exactly what a *term* is, using another non-terminal known as a *factor*, which is described in the final line. The symbols such as '+' and '-', and tokens such as **number** directly represent part of the input being examined.

Another way of looking at the grammar is from the bottom upwards. The smallest entity (or compilation unit as they are called), is a *factor*. This can consist of either a single number, a bracketed expression, or another factor preceded by a minus sign (for negation). At the next level up, a *term* is made up of a single *factor*, or a *term* followed by a '*' followed by a *factor*, or a *term* a '/' and then a *factor*. This production is mutually recursive, as the non-terminal *term* appears in the productions for itself. It should be possible to see that this definition for a *term* allows for an arbitrary number of *factors* separated by either '*'s or '/'s. If this is not obvious, consider starting with a *term*. If this is made up of a single *factor*, then the recursion stops before it has started. On the other hand, if we replace the *term* by the production '*term* * *factor*', we then have to replace the *term*. If we do so with a single *factor* then the recursion stops and we end up with '*factor* * *factor*'. If instead, we replace *term* by '*term* * *factor*' we get '*term* * *factor* * *factor*'. We can continue this recursion indefinitely, building up the string to the left, until we 'break-out' of the recursion. A similar argument applies to the productions for the non-terminal *expr*.

You may have spotted by now that the way an expression is split into *terms* and *factors* is related to the recognised operator precedence. For example, the productions for a *term* will group together strings of multiplications and/or divisions, while the productions for an *expr* group together addition and subtraction of *terms*. Indeed, this grammar was designed by considering the operator precedences, and having an individual non-terminal for each level.

To demonstrate that our grammar does indeed work, consider the expression:

(1+2)*3

We will start with the non-terminal *expr*, and at each stage replace a single non-terminal by one of its possible productions, as given above.

1. *expr*
2. *term*
3. *term* * *factor*
4. *factor* * *factor*
5. (*expr*) * *factor*
6. (*expr* + *term*) * *factor*
7. (*term* + *term*) * *factor*
8. (*factor* + *term*) * *factor*
9. (*factor* + *factor*) * *factor*
10. (1 + *factor*) * *factor*
11. (1 + 2) * *factor*
12. (1 + 2) * 3

In the above example, the values of the numbers involved have been retained so that their origins are obvious. However, in practice they would all be represented by the single token **number**, as returned by the lexical analyser.

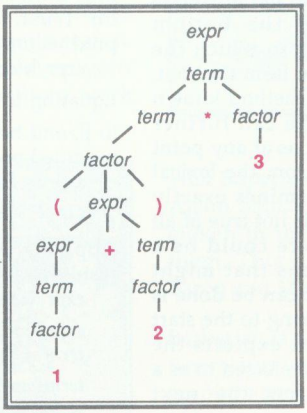


Figure 1. Parse tree for expression (1+2)*3. The leaf nodes which spell out the original expression are shown in colour

Between one line and the next, one, and only one, non-terminal has been replaced. For example, between the first and second lines, the production:

expr := *term*

has been applied, while between lines 4 and 5, the production:

factor := (*expr*)

has been applied. The process of using a grammar to derive a particular construct for a language is known as *parsing*, and another name for a syntax analyser is a *parser*. One common way of representing a derivation such as the one above is to use a so-called *parse* tree. The tree for this particular derivation is shown in figure 1. The root node of a parse tree represents the starting non-terminal of the grammar, *expr* in this case, and for any node except a leaf, the children of that node, read from left to right, represent the chosen production of the non-terminal at that node, as defined by our grammar. The leaf nodes, when read from left to right, yield the expression that has been matched.

One point to note at this stage is that at each point in the derivation of the expression, the left-most non-terminal has been replaced. For example, between steps four and five above, either of the *factors* could have been replaced, but the left-hand one was chosen. Such a derivation is called a *Left-most Derivation*. It might not seem important in which order the productions are applied, because they will all be applied in the end. However, as we shall see when we come to perform the actual compilation, keeping with a left-most derivation will ensure that the normal left-to-right associativity of arithmetic expressions is retained. In fact, in compiler jargon, it is normally assumed that any derivation will be left-most, and so this is not specified explicitly.

IMPLEMENTING THE SYNTAX ANALYSER

Now we have a grammar that represents expressions, we need to consider how to implement the syntax analyser using this grammar. There are two basic methods of parsing that can be used by a syntax analyser - *bottom up* parsing, in which the parse tree is effectively constructed from the bottom upwards, and *top down* parsing in which the reverse is done, building the tree from the root. We are going to use a top down method known as *recursive descent* parsing. We can further simplify matters by realising that as at any point in the string of input symbols from the lexical analyser, the next symbol determines exactly which production to apply. This is not true of all grammars - in some cases there could be a number of possible productions that might match at each stage, and all that can be done is to try each one in turn, backtracking to the start if we are wrong. A parser that exploits the uniqueness of each production is referred to as a *predictive parser*, as it predicts the next production to use at each stage.

The way in which our parser will be constructed is as a number of procedures, one for each non-terminal. To decode an expression, the parent program will call the procedure that handles the non-terminal *expr*. Unfortunately, we encounter a problem at this point. What happens is that the first production considered for the non-terminal *expr* is *'expr + term'*. Therefore, the procedure that handles *expr* will immediately call itself to try and match another *expr*. This will be repeated ad-infinitum and the entire program will hang. This problem will occur with any grammar in which the left-most non-terminal of a production for a particular non-terminal is itself, so for example, the same problem would be encountered with the procedure for *term*, which would call itself repeatedly. This is known as *left recursion*, and luckily can be avoided.

In order to eliminate left recursion, we have to rewrite our grammar in a way which solves the problem, but doesn't alter the meaning of the grammar at all. This can be done by modifying the offending productions according to a standard rule. This rule states that for any production of the form:

$A := A\alpha \mid \beta$

where A is a non-terminal, a and b and are string of non-terminals and symbols, the production can be replaced by:

$A := \beta R$

$R := \alpha R \mid \varepsilon$

R is a new non-terminal, and ε represents a null string (i.e. nothing). The proof of this rule can be a little difficult to follow, so you are asked to take it on trust. As an example, consider our productions:

expr := *expr* + *term* | *expr* - *term* | *term*

Equating to the rule, *expr* corresponds to A, term to β, and both + *term* and - *term* to α. Hence, the modified productions become:

expr := *term expr2*

expr2 := + *term expr2* | - *term expr2* | ε

Applying these changes to our entire grammar results in a new augmented grammar, thus:

expr := *term expr2*

expr2 := + *term expr2* | - *term expr2* | ε

term := *factor term2*

term2 := * *factor term2* | / *factor term2* | ε

factor := **number** | (*expr*) | - *factor*

While this new grammar does eliminate left recursion, it is longer and not so easy to follow. For this reason, it is conventional to design a grammar ignoring left recursion, and then eliminate it using the standard rule.

Next month we will look at a technique called *Syntax Directed Translation* which allows our parser not only to recognise expressions, but also to generate the machine code to implement them at the same time. We will combine these techniques to produce a working compiler program.

Games Review

by Peter Rochford

This year sees the usual sprinkling of new games releases in time for the Christmas spending spree.

Superior Soccer is the latest from Superior, and is both an arcade and football management game. You can choose to play either type of game, or play both together if you so desire.

In the arcade game, you have a screen view of the pitch from above in the form of a window that scrolls in all directions. You can play against the computer, or against another player. There are all the expected features such as dribbling, passing, tackling etc. In the bottom corner of the screen is a commentator who gabbles endlessly just like they do on TV.

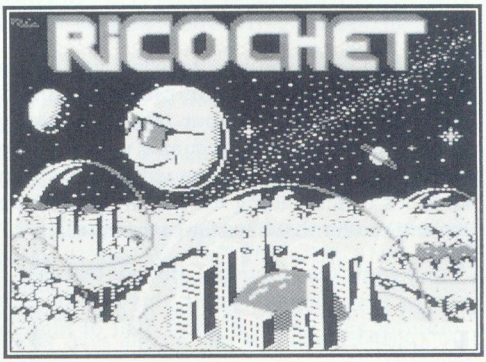
The football management game involves managing your own team throughout a season. You can buy and sell players, build up your team and generally use your skills to work your way up through the divisions.

Superior Soccer

Overall, this is a splendid game to play with nice graphics in the arcade part and is sure to please those who are interested in football, I would think.

Next off from Superior is Ricochet, an arcade adventure that boasts a mere 330 screens!

In Ricochet you control SPRAT, a robot time traveller, in his quest to collect the five hour glasses from the five worlds of Ricochet. There are the usual puzzles to solve along the way, and it all moves along very fast with some nice graphics and good sound. This one gets my vote as one of the better releases around at the moment for the Beeb.



Ricochet

Well Superior just cannot resist another Play It Again Sam, can they? Here we are at number eleven in the long line of compilations from them. This one features Barbarian, Pipeline, Baron and Monsters.

Barbarian is a sword fighting game with one or two player options. It features some excellent graphics and plenty of action with an added bit of humour to make it all the more enjoyable.

Pipeline is an arcade adventure with the usual host of puzzles to solve and a horde of nasties to avoid. The game features a smooth four way scrolling screen, nice graphics, but some nauseating background music.

Baron, the third game on the disc, has never been released before. It is an arcade adventure much like many others but not as good as most. Least said about this one the better. Last on the disc is the old Acornsoft classic Monsters. Ah,

how the old memories flood back! The first time I ever clapped eyes on a Beeb, someone was playing this game on it. I bought the computer and the game on the spot! Monsters is a version of the Apple game Panic where you run along walls and up and down ladders and dig holes to trap the monsters that chase you. Simple but addictive fun.

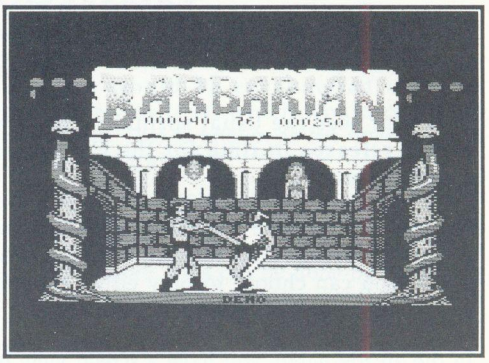
All in all P.I.A.S. 11 is a pretty good compilation apart from Baron. Recommended.

Finally, we have a couple of releases from Audiogenic who had the audacity to send them to me on cassette. Grrrr!! Well, really!

Blast! is an arcade adventure where you control a spaceship inside a complex network of underground caverns. Your task is to destroy the alien defences and make your escape. Naturally, you have to cope with attacks from all kinds of marauding nasties, not to mention the anti-matter that lines the cavern walls.

A nice game this, I thought, but not that easy. If you've ever played Thrust, this has much in common and needs a good deal of manual dexterity.

Also from Audiogenic is Fab Four Vol.1, their answer to Superior's Play It Again Sam. This compilation comprises Peter Scott's Thunderstruck and Omega Orb, both arcade adventures that feature some excellent programming and lots of challenging puzzles.



Barbarian from Play it Again Sam II

Along with these, come two re-hashes of Gary Partis' Psychastria and Sphere Of Destiny. I always found the original versions of these difficult to play. These mark 2 versions are just as awkward. Not my cup of tea I must say.

Fab Four is a reasonable compilation, but I would recommend it to the more seasoned games player.

That's it for 1989.
And a very Merry Christmas to you all!

**All Superior Software games are available from BEEBUG.*

B

Points Arising....Points Arising....Points Arising....Points Arising....

INSTANT PUBLISHING (BEEBUG Vol.8 No.6)

At the end of line 1090 of the GrabIt program, the 'GOTO980' should be replaced with 'GOTO 1160'.

ACES HIGH CARD GAME (BEEBUG Vol.8 No.4)

The amendment in last month's 'Points Arising' was incomplete. Line 1995 should read:

```
1995 IF Y%>20 Y1%=20 ELSE Y1%=Y%
```

FONT DESIGNER (PART 2) (BEEBUG Vol.8 No.5)

Attempting to use the Print option will result in an error. To cure this, replace 'PROCpt_ch' in lines 2110 and 2470 by 'PROCpt_ch(L%)'.

In addition, line 580 as given in part one should be changed to read as follows when part two is incorporated:

```
580 IF option%=0 OR option%>8 THEN 710
```

Our apologies for these and other recent lapses. We expect to return to our normal high standards from this issue.

B

Amateur Research (3)

John Belcher takes another irreverent look at the world of science.

ACTION AT A DISTANCE

This month we take our leave of astronomy, and have an important look at elementary physics.

Isaac Newton formulated the concept of *inverse-square-law* force as it applies to *gravitational force*. Andre-Marie Coulomb extended this idea to include *electrical force* and *magnetic force*. Michael Faraday dangled his watch chain, and made it four-of-a-kind with *electromagnetic force*.

And today, we are still none the wiser than was Newton 350 years ago, in that we are unable to solve the *many-body problem* involving three or more sources of force.

The difficulty is that modern science is peopled by what I call 'mechanics' and 'opticians'. Over the years, the former have been immersed in mechanical force. So much so, that they tend to treat every other force as if it was an exercise in mechanical engineering. Mechanical force is the force of direct action, the direct action between ropes and pulleys; pinions and gear wheels; one girder of a bridge and its neighbour; a cannon and its shell; and between a rocket and its propellant. The resultant force - where more than two sources of force are involved - can be resolved using vector analysis.

In complete contrast, inverse-square-law forces - of Newton et al - are the result of action at a distance - action involving space. Because of this, such forces cannot be resolved by the application of vector analysis. We must look to another form of solution.

Which is why, dear reader, we initially have to go back to space and space systems, because it is there - and nowhere else - where such a solution is to be found.

So cheer up! We're already halfway towards finding the answer!

MULTI-TO-SINGLE-ELEMENT FORCE TRANSFORM

Last month we dealt with the multi to single body space-system Transform. And just as Newton's two-body space-system - the Earth-Moon system - gave us the simple force solution, so too will our many-body system - the Solar System - lead to our understanding of many-body force problems. Messrs Coulomb and Faraday, take note.

We have to make some amendments, first of all. A space-system is itself a self contained, self maintained, dynamic, orbital vortical system, very different from a situation involving a few massive bodies, electric charges, or magnetic poles.

To start with, we replace the central or primary body of the system, M0, with a reference element, A0, where A0 has unit mass, unit charge, or unit pole as the case may be. The clever ones among you will now exclaim, "Ah! We're measuring field strength, not force!". Yes, in a loose kind of fashion, we are. But don't take this term too seriously. It has its roots in electrical engineering, not mechanical engineering. And according to my *Revised System of Dimensions* the connection is not a valid one.

Then again, we enter the situation with a reference vector, RE. This has the advantage that we can rotate it around the reference element and so obtain a polar diagram of force. That said, I haven't the foggiest idea what it would look like, or what we would call it, in three dimensions!

We then identify each secondary element, Ax, with its associated direction cosine, COS(thetax), where *thetax* is the angle subtended by Ax to the reference vector, RE. Thus each element is in the form, Ax*COS(thetax).

It follows that the resulting source magnitude is given by:

```
AE = A1*COS(theta1) + A2*COS(theta2)
     + ... + AN*COS(thetaN)
```

Rx, the distance between A0 and Ax, needs to be 'weighted' according to the magnitude of Ax*COS(thetax), each element being in the form:

```
Rx^((Ax*COS(thetax))/AE)
```

from which, the resultant distance between A0 and AE, which is effectively the reference vector itself, is:

```
RE = R1^((A1*COS(theta1))/AE)
     * R2^((A2*COS(theta2))/AE)
     * ... * RN^((AN*COS(thetaN))/AE)
```

Ultimately, the resultant force is given by:

```
Fn=k*A0*AE/RE^2
```

You may wish to determine what the single-element situation becomes under these circumstances! Our polar diagram, once again? To simplify matters considerably, we make the constant of proportionality, k=1, and A0=unity=1, and so Fn=AE/RE^2.

And at this point, dear reader, we are now in a position to determine the secrets of the forces of Nature!

MULTIPLE POINT-SOURCES IN ONE-DIMENSION

In 1972, when I had got as far as this in my research, I learned much about the Force Transform by solving multiple-force situations in one dimension using a Faber & Faber 5" slide-rule.

You, dear reader, are invited to repeat the experience, this time using the Basic program listed here, BPROG3. This will enable you to get the feel of the Transform that much the better. And if, eventually, you discuss your results with sceptical friends and colleagues, maybe you will be encouraged to investigate for yourself some new features which you may chance upon.

Key in the program and save it. When the program is run it asks for the number of dimensions, d%, in this case 1, followed by the value of N, the number of elements or point-sources to be processed, and then, in turn, for each element, the source magnitude, A, and its distance along the X-axis. This is arranged in lines 100-160. Lines 180-200 take care of the necessary calculation, and output the values of AE, RE, and Fn. All the clever work is done by PROCcalc1 - which determines the value of AE, and PROCcalc2 - which determines the value of RE.

Although it is possible to identify elements in the form A,R/theta,phi, i.e. using polar co-ordinates, most situations are best defined and simulated using rectangular co-ordinates, A,X,Y,Z. That said, in the one-dimensional examples that follow, we will use the convention, d%:N:A1,X1; A2,X2; A3,X3; etc, to denote the necessary values to be input.

1. FOR ANY GIVEN DISTANCE, *AE* IS THE ALGEBRAIC SUM OF THE MAGNITUDES OF THE ELEMENTS CONCERNED. Input 1:1:10,10 and AE=10, RE=10, F1=0.1. Input 1:1:5,10 and AE=5, RE=10, F1=5E-2. Input 1:1:-8,10 and AE=-8, RE=10, F1=-8E-2. Then input 1:3:10,10; 5,10; -8,10 and AE=7, RE=10, and F3=6.99999999E-2.

2. FOR ANY GIVEN LINEAR UNIFORM ARRAY OF ELEMENTS, RE IS THE GEOMETRIC MEAN OF THE INDIVIDUAL DISTANCES. Input 1:5:1,102; 1,101; 1,100; 1,99; 1,98 and AE=5, RE=99.989999, and F5=5.00100027E-4. Close to, however, proximity-profile effect produces the 'spark-gap' phenomenon! Input 1:5:1,5; 1,4; 1,3; 1,2; 1,1 whence AE=5, and RE<>3 but RE=2.60517109. Thus F5<>0.55555555 but F5=0.7367115995, some 50% greater.

3. IN ANY FORCE CONFIGURATION, THE SOURCE OF GREATEST MAGNITUDE EXERTS THE GREATEST INFLUENCE, *IRRESPECTIVE OF ITS DISTANCE*. Input 1:3:10,10; 100,100; 1000,1000 From which AE=1110, effectively the magnitude of the greatest source, and RE=779.636013 effectively the distance of the greatest source. Which is perhaps why the Sun exerts a greater effect on the Earth than a lump of rock on its surface! In a manner of speaking, that is!

4. CERTAIN FORCE CONFIGURATIONS GIVE RISE TO FORCES OF EXTREME MAGNITUDE. Input 1:2:-21,2; 20,10. From which AE=-1, RE=2.097152E-14 and F2=-2.27373675E27!

5. CERTAIN FORCE CONFIGURATIONS GIVE RISE TO FORCES OF INFINITE MAGNITUDE. Input 1:2:-20.5,2; 20,10. From which AE=-0.5, RE=zero, and F2=infinity. Sorry the program crashed!

Over the years, I have come to refer to the extreme and infinite forces of (4) and (5) as cataclysmic force, largely because the term 'infinite' is impractical in the computing sense.

MULTIPLE POINT-SOURCES IN TWO DIMENSIONS

BPROG3 can also be used for solving problems in *two* dimensions by inputting data in the form d%:N:A1,X1,Y1; A2,X2,Y2; etc (i.e. number of dimensions followed by number of elements, and then three values - A, X, Y - for each element).

It is, of course, possible to play cat's cradle with the point-sources positioned all over the two dimensional grid, but the opportunity now presents itself to analyse the force characteristics of non-point source bodies. This leads us to the simulation of such bodies in two dimensions, the resulting simulations effectively describing arrays of elements.

In two dimensions, such arrays in turn describe areas of defined shape or profile. The shapes or profiles of fundamental interest are those which are symmetrically arranged with respect to the centre, e.g. discs and rings. Of less regular profile, are the open-ended horseshoe arrays, which have a special significance where cataclysmic force is concerned.

The simulation of arrays of regular profile has, over the years, tended to follow a definite pattern. In a two dimensional array, only one quadrant need be defined, its data being altered in sign to represent the remaining quadrants. Fig-3.1 shows the simulation of a simple disc. Simple, because although it is of uniform 'density', the simulation is necessarily 'rough'. Ideally, an infinite number of elements should be employed, but in practice a 1000-element array is the practical optimum. Imagine keying in that amount of data! For certain investigations e.g. into what is now known as the 'fifth' and 'sixth' forces, a pseudo-infinite array can be employed, in which individual elements are broken down into sub-elements of 1/100-th the normal value.

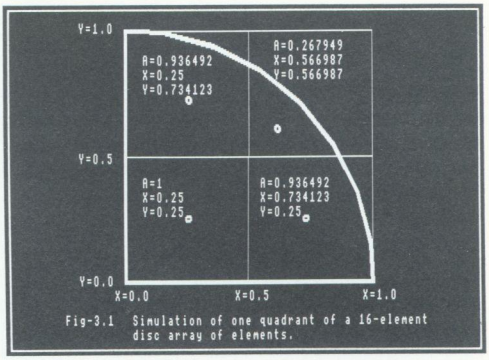


Fig. 3.1

In the case of the simple disc, the quadrant is divided into unit areas, or parts of unit area. At the C of G - or centre of force - of each unit area is a point-source representing unit mass, unit charge, or unit pole, as the case may be. In areas of less than unit size, the value of the point-source is adjusted accordingly. It will be seen that no point-source lies near the X-axis or the Y-axis, and this has the advantage that the trigonometrical functions of small angles need not be calculated.

When A0, the reference element, is located within the perimeter of such a disc, the value of RE=1.0. It will be seen later, that RE thence effectively becomes the unit of length of the array.

For the time being, it is perhaps easier to investigate the force characteristics of the simple arrays shown in Fig-3.2.

6. The ***Broadside*** array has characteristics the very opposite of the ***Endfire*** array given in (2). These descriptions are, of course, lifted from antenna theory in radio engineering. Whereas the ***Endfire*** array simulates the 'spark-gap' effect, the ***Broadside*** array suggests the electric field obtained from a 'flat plate'. Input 2:6:1,1,0.5; 1,1,1.5; 1,1,2.5; 1,1,-0.5; 1,1,-1.5; 1,1,-2.5. It will be seen that instead of AE=6, RE=1, and F6=6, we find that AE=3.64103613,

RE=1.54719684, and F6=1.52101623. Thus in this instance we get a 75% reduction in force, well in agreement with observations!

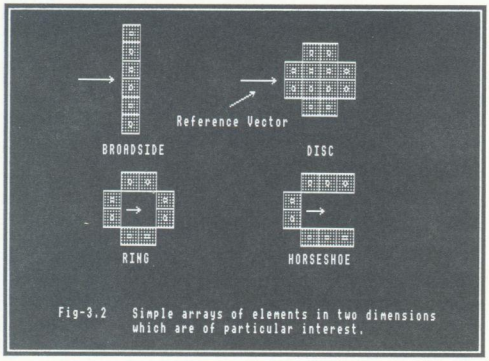


Fig-3.2 Simple arrays of elements in two dimensions which are of particular interest.

Fig. 3.2

7. The *Disc* array shown is a simpler version of Fig-3.1. To see the proximity effect at the perimeter of a DISC, input 2:12:1,2.5,0.5; 1,2.5,-0.5; 1,3.5,0.5; 1,3.5,-0.5; 1,2.5,1.5; 1,2.5,-1.5; 1,1.5,0.5; 1,1.5,-0.5; 1,1.5,1.5; 1,1.5,-1.5; 1,0.5,0.5; 1,0.5,-0.5. We would expect the result to read AE=12, RE=2, and F12=3. What we get is AE=10.3818399, RE=2.0805383, and F12=2.39491674, 20% lower than the anticipated value. This is the point of maximum proximity effect, the cause of the 'fifth' and 'sixth' forces!

8. The significant feature of the *Ring* array is the value of RE that results when the reference element is near the centre of the ring. Input 2:8:1,1.51,0.5; 1,1.51,-0.5; 1,-1.49,0.5; 1,-1.49,-0.5; 1,0.51,1.5; 1,0.51,-1.5; 1,-0.49,1.5; 1,-0.49,-1.5. You will see that RE=4.29801398, which expressed in terms of radius=1.5, is RE=2.86534265*radius and not UNITY. In an improved simulation, RE=EXP(1)=2.71828183. At which you will recognise one of my *three ubiquitous features!* That this relationship could be discovered by anyone determining the force characteristics of a Ring of Stones may have some historical significance!

9. The *Horseshoe* array - note the similarity to another ring of Stones the Stonehenge trilithon configuration - is a cataclysmic force array! Input 2:8:1,1.66,1.5; 1,1.66,-1.5; 1,0.66,1.5; 1,0.66,-1.5; 1,-0.34,1.5; 1,-0.34,-1.5; 1,-1.34,0.5; 1,-1.34,-0.5. Surprise! Surprise! We now find that AE=-2.65276427E-2, RE=1.0393029E-12, and F8=-2.45592094E22. You can, of course, easily improve on this result!

RETROSPECT AND PROSPECT

Well this month we have seen how physics can be put on the right lines when seen through the eyes of an astronomer rather than those of a mechanic. Next month we will take a look at megalithic physics, and see if Pre-Stone Age Man was as primitive as archaeologists would have us believe!

REFERENCES

(1) Belcher, J.C., 'The Multi to single element transform', Electronics & Wireless World, Vol. 93, No. 1614, (April, 1987), pp421-423.

```
   10 REM Program .>BPROG3
   20 REM Version B1.1
   30 REM Author  J.C.Belcher
   40 REM BEEBUG  December 1989
   50 REM Program subject to copyright
   60 :
  100 CLEAR
  110 MODE6:PRINT''"FORCE SITUATIONS IN
1 & 2 DIMENSIONS"
  120 INPUT'"Enter number of dimensions
- "d%:IF d%<>1 AND d%<>2 THEN GOTO 110
  130 INPUT'"Enter number of elements -
"N:DIM A(N),X(N),Y(N)
  140 FOR I%=1 TO N:PRINT'"ELEMENT NUMBE
R ";I%
  150 PRINT"Enter magnitude (A";I%;:INPU
T") = "A(I%):PRINT"Enter X-coordinate (X
";I%;:INPUT")  = "X(I%):Y(I%)=0:IF d%=2
THENPRINT"Enter Y-coordinate (Y";I%;:INP
UT")  = "Y(I%)
  160 NEXT:AE=0:RE=1
  170 FOR I%=1 TO N:PROCcalc1:NEXT
  180 FOR I%=1 TO N:PROCcalc2:NEXT
  190 PRINT'"AE : "AE'"RE : "RE'"F";N;"
: "AE/(RE*RE)
  200 PRINTTAB(0,24)"Press any key to co
ntinue";:G=GET:GOTO 100
  210 :
 1000 DEFPROCcalc1:R=SQR(X(I%)^2+Y(I%)^2
):a=A(I%)*(X(I%)/R):AE=AE+a:ENDPROC
 1030 DEFPROCcalc2:R=SQR(X(I%)^2+Y(I%)^2
):a=A(I%)*(X(I%)/R):R=R^(a/AE):RE=RE*R:E
NDPROC
```

```
2210 INY:INX:LDA(commandln),Y
2220 CMP#ASC".":BEQ spotfound
2230 CMP#space:BEQ endofip
2240 CMP#nl:BEQ endofip
2250 AND#&DF:CMP comtable,X:BEQ comlp
2260 JMP nxtcomlp
2270 .spotfound
2280 LDA comtable,X:BEQ nxtcom
2290 .sptlp
2300 INX:LDA comtable,X:BEQ jmpcom
2310 JMP sptlp
2320 .endofip
2330 LDA comtable,X:BEQ jmpcom
2340 .nxtcomlp
2350 LDA comtable,X:BEQ nxtcom
2360 INX:JMP nxtcomlp
2370 .nxtcom
2380 INX:INX:INX
2390 LDA comtable,X:CMP#&FF:BNE testcom
2400 .notus
2410 PLA:TAY:LDX &F4:LDA#4:RTS
2420 .jmpcom
2430 INY:INX:LDA comtable,X:STA hi
2440 INX:LDA comtable,X:STA lo:JMP(lo)
2450 .comtable
2460 EQUS"FBASIC":EQUB0:EQUW&008C
2470 EQUS"FTEXT":EQUB0:EQUW&0088
2480 EQUS"FPROCFN":EQUB0:EQUW&038C
2490 EQUS"LFROM":EQUB0:EQUW&068C
2500 EQUS"LPROC":EQUB0:EQUW&098C
2510 EQUS"LFN":EQUB0:EQUW&0C8C
2520 EQUS"RBASIC":EQUB0:EQUW&0F8C
2530 EQUS"RTEXT":EQUB0:EQUW&128C
2540 EQUS"SYSINF":EQUB0:EQUW&158C
2550 EQUS"VARLIST":EQUB0:EQUW&188C
2560 EQUS"FKDEFS":EQUB0:EQUW&1B8C
2570 EQUS"PCOMM":EQUB0:EQUW&1E8C
2580 EQUS"QCOMM":EQUB0:EQUW&218C
2590 EQUS"RCOMM":EQUB0:EQUW&248C
2600 EQUS"SCOMM":EQUB0:EQUW&278C
2610 EQUS"TCOMM":EQUB0:EQUW&2A8C
2620 EQUS"UCOMM":EQUB0:EQUW&2D8C
2630 EQUS"VCOMM":EQUB0:EQUW&308C
2640 EQUS"WCOMM":EQUB0:EQUW&338C
2650 EQUS"XCOMM":EQUB0:EQUW&368C
2660 EQUS"YCOMM":EQUB0:EQUW&398C
2670 EQUS"ZCOMM":EQUB0:EQUW&3C8C
2680 EQUW&FFFF
2690 :
2700 ]NEXT:ENDPROC
2710 :
2720 DEF PROCtokentable(addr%)
2730 ttable=&84F0
2740 FOR I%=0 TO&308
2750 I%?(code+&4F0)=I%?addr%
2760 NEXT:ENDPROC
2770 :
```

```
2780 DEF PROCassemble2
2790 p%=&8800:o%=code+&800
2800 FOR pass=4 TO7 STEP3
2810 P%=p%:O%=o%
2820 [OPT pass
2830 :
2840 .calltable
2850 JMP findtext
2860 JMP romexit:JMP escexit
2870 JMP initscanft:JMP scanprog
2880 JMP detokline:JMP prntline
2890 JMP prnttext:JMP notimp
2900 JMP initbptr:JMP initpptr
2910 JMP inittptr:JMP reinitbptr
2920 JMP initvector:JMP prnteoprog
2930 :
2940 .romexit
2950 LDX&F4:PLA:TAY:LDA#0:RTS
2960 :
2970 .initscanft
2980 LDA#scanfortextstr MOD256
2990 STA scantype
3000 LDA#scanfortextstr DIV256
3010 STA scantype+1:RTS
3020 :
3030 .scanprog
3040 .proglp LDY#1:LDA(bptr),Y:CMP#&FF
3050 BEQ proglpexit:LDY#4
3060 LDA#(nextline-1)DIV256:PHA
3070 LDA#(nextline-1)MOD256:PHA
3080 .scanline JMP(scantype)
3090 .nextline
3100 JSR reinitbptr:BIT&FF:BPLproglp
3110 JMP escexit
3120 .proglpexit RTS
3130 :
3140 .prntline TYA:PHA:LDX#0
3150 .prntlnloop
3160 LDA buffer,X:CMP#cr:BEQ prntlnexit
3170 JSR oswrch:INX:JMP prntlnloop
3180 .prntlnexit JSR osnewl:PLA:TAY:RTS
3190 :
3200 .prnteoprog JSR prnttext
3210 EQUS"Search complete":EQUWnl:RTS
3220 :
3230 .initvector
3240 LDA#prntline MOD256:STA action
3250 LDA#prntline DIV256:STA action+1
3260 RTS
3270 :
3280 .findtext
3290 JSR iptxtparam:JSR initbptr
3300 JSR initvector
3310 JSR initscanft:JSR scanprog
3320 JSR prnteoprog:JMP romexit
3330 :
3340 .scanfortextstr
```

```
3350 JSR detokline:LDY#5
3360 .scantextloop
3370 LDX#0:TYA:PHA:LDA buffer,Y
3380 CMP#nl:BEQ movetonextline
3390 CMP ipbuff,X:BEQ charmatchloop
3400 PLA:INY:JMP scantextloop
3410 .charmatchloop
3420 INX:INY:LDA ipbuff,X:CMP#cr
3430 BEQ strmatch:CMP buffer,Y
3440 BEQ charmatchloop:PLA:TAY:INY
3450 JMP scantextloop
3460 .strmatch STYex:PLA:TAY:INY
3470 LDA#(nextscantext-1)DIV256:PHA
3480 LDA#(nextscantext-1)MOD256:PHA
3490 JMP(action)
3500 .nextscantext JMP scantextloop
3510 .movetonextline PLA:RTS
3520 :
3530 .initbptr
3540 LDA page:STA bptr+1
3550 LDA#0:STA bptr:RTS
3560 .initpptr
3570 LDA #ipbuff MOD256:STA pptr
3580 LDA #ipbuff DIV256:STA pptr+1:RTS
3590 .reinitbptr
3600 LDY#3:LDA(bptr),Y
3610 CLC:ADCbptr:STAbptr
3620 LDA#0:ADC bptr+1:STA bptr+1:RTS
3630 :
3640 .iptxtparam
3650 LDA(commandln),Y:INY:CMP#space
3660 BEQ iptxtparam:CMP#nl:BEQ ipterr
3670 STA ipdelim:DEY:LDX#&FF
3680 .tparamlp
3690 INY:INX:LDA(commandln),Y
3700 STA ipbuff,X:CMP ipdelim
3710 BEQ tparamend:CMP#nl:BNE tparamlp
3720 .ipterr LDA#0:STA&100
3730 LDA#105:STA&101:LDX#0
3740 .ipterrlp
3750 LDA errmess,X:STA &102,X:INX:INY
3760 CMP#0:BNE ipterrlp:JMP &100
3770 .errmess EQUS"Delimiters?"
3780 EQUB0
3790 .tparamend
3800 LDA#nl:STA ipbuff,X:RTS
3810 :
3820 .detoklnno
3830 LDA#1:ORAflags:STA flags
3840 LDY#1:LDA(bptr),Y:STA hi
3850 INY:LDA(bptr),Y:STA lo
3860 :
3870 .convhextodec TXA:PHA
3880 LDY#8:LDA#&BF:AND flags:STAflags
3890 .digloop LDX#ASC"0"
3900 .subloop SEC:LDA lo:SBC powers,Y
3910 PHA:LDA hi:SBC powers+1,Y
3920 BCC storedig:STA hi:PLA:STA lo
3930 INX:BCS subloop
3940 .storedig PLA:TXA:CMP#ASC"0"
3950 BEQ zero:LDXay:STA buffer,X:INCay
3960 LDA flags:ORA#&40:STA flags
3970 .nextdigit DEY:DEY:BPL digloop
3980 .chdexit PLA:TAX:RTS
3990 LDA#&BF:AND flags:STAflags
4000 .zero BIT flags:BVC storespc:LDXay
4010 STA buffer,X:INCay:JMP nextdigit
4020 .storespc LDA#1:BIT flags
4030 BEQ nextdigit:LDA#space:LDXay
4040 STA buffer,X:INCay:JMP nextdigit
4050 .powers EQUW1:EQUW10:EQUW100
4060 EQUW1000:EQUW10000
4070 :
4080 .findtoknstring \token in "token"
4090 JSR inittptr:LDY#0
4100 .tokenloop
4110 LDA(tptr),Y:BMI tokenfnd
4120  INY:JMP tokenloop
4130 .tokenfnd
4140 CMP token:BEQ tknmatch
4150 CMP#&D3:BEQ notknmatch
4160 CMP#numtoken:BEQ notknmatch
4170 .nexttoken
4180 INY:INY:TYA
4190 CLC:ADC tptr:STA tptr
4200 LDA#0:ADC tptr+1:STA tptr+1
4210 LDY#0:JMP tokenloop
4220 .tknmatch DEY:RTS \ length in Y
4230 .notknmatch LDY#0:RTS
4240 .inittptr
4250 LDA#ttable MOD256:STA tptr
4260 LDA#ttable DIV256:STA tptr+1:RTS
4270 :
4280 .detokline
4290 STYwy:TYA:PHA:LDX#0:STXay
4300 JSR detoklnno:LDY#3:LDXay
4310 .detokbas
4320 INY:CPYwy:BNE detokbas2:STXex
4330 .detokbas2
4340 LDA(bptr),Y:CMP#nl:BEQ dtexit
4350 JSR movech:CMP#quote:BEQ quotelp
4360 JMP detokbas
4370 .movech
4380 CMP#0:BMI movetokn
4390 STA buffer,X:INX:RTS
4400 .movetokn
4410 CMP#numtoken:BEQ numtkn:STA token
4420 TYA:PHA:JSR findtoknstring:LDY#0
4430 .movetoknloop LDA(tptr),Y
4440 BMI mtokenexit:STA buffer,X:INX
4450 INY:JMP movetoknloop
4460 .mtokenexit PLA:TAY:LDA#0:RTS
4470 .numtkn
4480 INY:LDA(bptr),Y:ASL A:ASL A
```

```
4490 PHA:AND#&C0:INY:EOR(bptr),Y
4500 STA lo:INY:PLA:ASLA:ASL A
4510 EOR(bptr),Y:STA hi:LDA#&FE
4520 AND flags:STA flags:TYA:PHA:STXay
4530 JSRconvhextodec:LDXay:PLA:TAY:RTS
4540 .dtexit LDA#nl:STA buffer,X
4550 PLA:TAY:RTS
4560 .quotelp INY:LDA(bptr),Y
4570 CMP#nl:BEQdtexit:STAbuffer,X:INX
4580 CMP#quote:BNE quotelp:JMP detokbas
4590 :
4600 .prnttext
4610 PLA:STA lo:PLA:STA hi:LDY#1
4620 .plp
4630 LDA(lo),Y:JSR osasci:INY:CMP#0
4640 BNE plp:CLC:TYA:ADC lo:STA lo
4650 LDA#0:ADC hi:STA hi:JMP(lo)
4660 :
```

```
4670 .escexit
4680 LDA#&7E:JSRosbyte:LDA#0:STA &100
4690 LDA#17:STA &101:LDX#0
4700 .esclp LDA escape,X:STA &102,X
4710 INX:CMP#0:BNE esclp:JMP &100
4720 .escape EQUS"Escape":EQUB0
4730 :
4740 .notimp JSR prnttext
4750 EQUS"Not implemented":EQUWnl
4760 JMP romexit
4770 :
4780 ]NEXT:ENDPROC
4790 :
4800 DEF PROClinks
4810 P%=start+size:O%=code+size
4820 FORI%=0 TO20
4830 [OPT7:JMP notimp:]
4840 NEXT:ENDPROC
```

Solids of Revolution (continued from page 34)

```
ireframe model."
 1250 COLOUR1:PRINT"2";:COLOUR3:PRINT" A
lternate facets."
 1260 PRINT'"Storage:"'
 1270 COLOUR1:PRINT"7";:COLOUR3:PRINT" S
ave profile."
 1280 COLOUR1:PRINT"8";:COLOUR3:PRINT" L
oad new profile."
 1290 PRINT:COLOUR1:PRINT"Escape";:COLOU
R3:PRINT" to return to this page."
 1300 ENDPROC
 1310 :
 1320 DEF PROCedit
 1330 LOCAL flag%,a%,one%,two%,three%,fi
rst%
 1340 flag%=FALSE
 1350 REPEAT
 1360 GCOL0,1:MOVE600-px%(1),py%(1):FOR
a%=2 TO px%(0):DRAW600-px%(a%),py%(a%):N
EXT
 1370 GCOL0,0
 1380 one%=py%(0)-1:IF one%<1 one%=1
 1390 two%=py%(0)
 1400 three%=py%(0)+1:IF three%>px%(0) t
hree%=px%(0)
 1410 MOVE600-px%(one%),py%(one%):DRAW60
0-px%(two%),py%(two%):DRAW600-px%(three%
),py%(three%)
 1420 :
 1430 *FX4,1
 1440 GCOL3,3:first%=TRUE
 1450 *FX138,0,32
 1460 REPEAT
 1470 g%=GET
 1480 IF first%=FALSE MOVE600-px%(one%),
```

```
py%(one%):DRAW600-px%(two%),py%(two%):DR
AW600-px%(three%),py%(three%)
 1490 IF first%=TRUE first%=FALSE
 1500 IF g%=136 px%(py%(0))=px%(py%(0))+
4:IF INKEY(-1)px%(py%(0))=px%(py%(0))+12
 1510 IF g%=137 px%(py%(0))=px%(py%(0))-
4:IF INKEY(-1)px%(py%(0))=px%(py%(0))-12
 1520 IF g%=138 py%(py%(0))=py%(py%(0))-
4:IF INKEY(-1)py%(py%(0))=py%(py%(0))-12
 1530 IF g%=139 py%(py%(0))=py%(py%(0))+
4:IF INKEY(-1)py%(py%(0))=py%(py%(0))+12
 1540 IF g%=49 OR g%=50 OR g%=55 OR g%=5
6 flag%=TRUE
 1550 one%=py%(0)-1:IFone%<1one%=1
 1560 two%=py%(0)
 1570 three%=py%(0)+1:IF three%>px%(0) t
hree%=px%(0)
 1580 MOVE600-px%(one%),py%(one%):DRAW60
0-px%(two%),py%(two%):DRAW600-px%(three%
),py%(three%)
 1590 UNTIL g%=65 OR g%=97 OR g%=90 OR g
%=122 OR g%=75 OR g%=107 OR flag%=TRUE
 1600 :
 1610 IF g%=65 OR g%=97 py%(0)=py%(0)+1:
IF px%(0)<py%(0) py%(0)=px%(0)
 1620 IF g%=90 OR g%=122 py%(0)=py%(0)-1
:IF py%(0)=0 py%(0)=1
 1630 IF g%=75 OR g%=107:IF px%(px%(0))<
>0:IF py%(px%(0))<>1024:IF px%(0)<19 px%
(0)=px%(0)+1:py%(0)=px%(0):py%(px%(0))=1
024
 1640 UNTIL flag%
 1650 ENDPROC
 1660 :
 1670 DEF PROCwireframe
```

```
 1680 LOCAL I,a%
 1690 PROCinitrotation(0,90,-20)
 1700 GCOL0,3
 1710 FOR I=0 TO 2*PI STEP 0.4
 1720 a%=1:PROCmove(px%(a%)*COS(I),py%(a
%)-500,px%(a%)*SIN(I))
 1730 FOR a%=1 TO px%(0)
 1740 PROCdraw(px%(a%)*COS(I),py%(a%)-50
0,px%(a%)*SIN(I))
 1750 NEXT:NEXT
 1760 a%=1:PROCmove(px%(a%)*COS(I),py%(a
%)-500,px%(a%)*SIN(I))
 1770 FOR a%=1 TO px%(0)
 1780 PROCmove(px%(a%)*COS(0),py%(a%)-50
0,px%(a%)*SIN(0))
 1790 FOR I=0 TO 2*PI STEP 0.2
 1800 PROCdraw(px%(a%)*COS(I),py%(a%)-50
0,px%(a%)*SIN(I))
 1810 NEXT I
 1820 PROCdraw(px%(a%)*COS(0),py%(a%)-50
0,px%(a%)*SIN(0))
 1830 NEXT
 1840 PRINT"SPACE TO CONTINUE":REPEAT UN
TIL GET=32
 1850 ENDPROC
 1860 :
 1870 DEF PROCfacet
 1880 LOCAL d%,a%,e%
 1890 PROCinitrotation(0,90,-20):c=PI/8
 1900 d%=1:FOR a%=1 TO 8:e%=d%:FOR b%=2
TO px%(0):PROCrectangle(d%,3):IF d%=1 d%
=2 ELSE d%=1
 1910 NEXT:IF e%=1 d%=2 ELSE d%=1
 1920 NEXT:c=-c
 1930 d%=2:FOR a%=1 TO 8:e%=d%:FOR b%=2
TO px%(0):PROCrectangle(d%,3):IF d%=1 d%
=2 ELSE d%=1
 1940 NEXT:IF e%=1 d%=2 ELSE d%=1
 1950 NEXT
 1960 PRINT"SPACE TO CONTINUE":REPEAT UN
TILGET=32
 1970 ENDPROC
 1980 :
 1990 DEF PROCrectangle(c1%,c2%)
 2000 GCOL0,c1%
 2010 PROCmove(px%(b%-1)*COS(c*(a%-1)),p
y%(b%-1)-500,px%(b%-1)*SIN(c*(a%-1)))
 2020 PROCmove(px%(b%-1)*COS(c*(a%)),py%
(b%-1)-500,px%(b%-1)*SIN(c*(a%)))
 2030 PROCplot85(px%(b%)*COS(c*(a%-1)),p
y%(b%)-500,px%(b%)*SIN(c*(a%-1)))
 2040 PROCplot85(px%(b%)*COS(c*(a%)),py%
(b%)-500,px%(b%)*SIN(c*(a%)))
 2050 GCOL0,c2%
 2060 PROCmove(px%(b%-1)*COS(c*(a%)),py%
(b%-1)-500,px%(b%-1)*SIN(c*(a%)))
 2070 PROCdraw(px%(b%-1)*COS(c*(a%-1)),p
y%(b%-1)-500,px%(b%-1)*SIN(c*(a%-1)))
 2080 PROCdraw(px%(b%)*COS(c*(a%-1)),py%
(b%)-500,px%(b%)*SIN(c*(a%-1)))
 2090 PROCdraw(px%(b%)*COS(c*(a%)),py%(b
%)-500,px%(b%)*SIN(c*(a%)))
 2100 PROCdraw(px%(b%-1)*COS(c*(a%)),py%
(b%-1)-500,px%(b%-1)*SIN(c*(a%)))
 2110 ENDPROC
 2120 :
 2130 DEF PROCinitrotation(A,B,C)
 2140 LOCAL a,b,c
 2150 a=RAD(A):p=COS(a):q=SIN(a):r=-q:s=
p
 2160 b=RAD(B):t=COS(b):u=SIN(b):v=-u:w=
t
 2170 c=RAD(C):l=COS(c):m=SIN(c):n=-m:o=
l
 2180 ENDPROC
 2190 :
 2200 DEF PROCrotate
 2210 LOCAL xs,ys
 2220 xs=x:x=(xs*p)+(y*r):y=(xs*q)+(y*s)
 2230 xs=x:x=(xs*t)+(z*v):z=(xs*u)+(z*w)
 2240 ys=y:y=(ys*l)+(z*n):z=(ys*m)+(z*o)
 2250 PROCperspective(900,1)
 2260 ENDPROC
 2270 :
 2280 DEF PROCperspective(F,S)
 2290 z=z+1000:x=(F*x)/z:y=(F*y)/z
 2300 x=x*S:y=y*S:x=x+640:y=y+512
 2310 ENDPROC
 2320 :
 2330 DEF PROCmove(x,y,z):PROCrotate:MOV
Ex,y:ENDPROC
 2340 :
 2350 DEF PROCdraw(x,y,z):PROCrotate:DRA
Wx,y:ENDPROC
 2360 :
 2370 DEF PROCplot85(x,y,z):PROCrotate:P
LOT85,x,y:ENDPROC
 2380 :
 2390 DEF PROCsave_load(d$)
 2400 LOCAL f,a
 2410 PRINT'"Enter filename":INPUTf$
 2420 IF d$="S" f=OPENOUT f$ ELSE f=OPEN
IN f$
 2430 IF d$="L" AND f=0 PRINT"file not f
ound."'"Space to continue":REPEAT UNTIL
 GET=32:ENDPROC
 2440 FOR a=0 TO 20
 2450 IF d$="S" PRINT#f,px%(a),py%(a) EL
SE INPUT#f,px%(a),py%(a)
 2460 NEXT
 2470 CLOSE#f
 2480 ENDPROC
```

More hints for the Beeb selected by Mike Williams.

FINDING THE COLOUR OF A POINT

John McFarlane

If you wish to find the colour of the pixel at the graphics cursor, but don't know the actual co-ordinates, try the following function:

```
1000 DEF FNcur_point
1010 X%=&10
1020 Y%=&3
1030 A%=&9
1040 CALL &FFF1:REM call osword
1050 =?&314
```

OSWORD 9 is the machine code equivalent of the POINT function, and requires a four-byte parameter block to read the cursor position. Fortunately, memory locations &310 to &313 hold the graphics cursor position, so we just need to direct OSWORD 9 to it (lines 1010, 1020).

MODE 0 SCREEN DUMP

Paul Holmes

The following routine provides an easy-to-use mode 0 screen dump to an Epson compatible printer for use in any program:

```
100 DEF PROCdump
110 LOCAL X%,Y%,L%
120 REM Disable Escape
130 *FX229,32
140 escape=FALSE
150 VDU2
160 VDU1,27,1,64,1,24
170 VDU1,27,1,51,1,21
180 FOR Y%=1023 TO 0 STEP -16
190 VDU1,27,1,90,1,128,1,7
200 X%=0
210 REPEAT
220 IF INKEY-113 THEN escape=TRUE
230 L%=0
240 IF POINT(X%,Y%)=1 L%=L%+192
250 IF POINT(X%,Y%-4)=1 L%=L%+48
260 IF POINT(X%,Y%-8)=1 L%=L%+12
270 IF POINT(X%,Y%-12)=1 L%=L%+3
280 VDU1,L%,1,L%,1,L%
290 X%=X%+2
300 UNTIL X%>1279
310 VDU1,10:REM line feed
320 IF escape THEN Y%=-16
330 NEXT Y%
340 VDU1,27,1,64,1,24:VDU3
350 REM restore Escape key
360 *FX229
370 ENDPROC
```

This is derived from a program by David Lowndes-Williams in Vol.6 No.6. It does not draw an accurate circle, but does give a similar aspect ratios to the screen, and is suitable for most straight line work. It has the advantage of reproducing text clearly, and is unaffected by graphics windows and other such complexities.

PROGRAMMING VIEWSTORE

Ashley Allerton

There is an apparent problem if you want to create a 'front-end' menu program to access ViewStore, as everything in memory is lost when you switch from Basic to ViewStore. A solution is to code the necessary instructions as one or more function key definitions, and then use *FX138 from within the program to cause the function key routine to be executed.

Here is an example, which prints out an up-to-date phone directory from a database which maintains a list of names, addresses, telephone numbers etc., and then returns the user to Basic:

```
  100 DEF PROCphone
  110 OSCLI("KEY9 LOAD ADDRESSES:MODE131
|MUTILITY SELECT|MF|M|MSURNAME|MCHRISTIA
N NAME|M|M|MUTILITY REPORT|MY|MP|MY|MPHO
NEDIR|M|M|M|M*KEY9|M")
  120 OSCLI("KEY8 *BASIC|M|M*MOUNT 0|MCH
AIN ""MENU""|M*KEY8|M")
  130 *FX138,0,137
  140 *FX138,0,136
  150 *STORE
  160 ENDPROC
```

Note that the last instruction of each function key definition cancels that definition. The above routine is just an example: you will need to create your own program and routines to suit the database that you have created.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

BBC B issue 7/Opus DDOS, Opus DS 80/40 DD, Voltmace joystick, Replay ROM fitted and several BBC books £375, 50 BBC games cassettes £80, 20 BBC games discs £70, or BBC with all games £475. Electron with 15 games & joystick £45. Tel. 061- 626 9170 eves.

View Printer Driver Generator £6, View manual £6, Viewsheet manual £6, all inc p&p. Tel. (0436) 72573.

Wordwise plus £15, Exmon II £12, both boxed, Printmaster ROM £12, Morley Teletext adaptor £45, View User Manual £3, Master Reference Manual £5, all perfect. Tel. (0704) 69650.

Master 128 £300, also books, genuine software, joystick, Teletex etc. Tel. for details. Tel. (04243) 4500.

Archimedes colour monitor, leads, instructions £130. NECP 2200 24pin printer, boxed, instructions, leads £180. Logistix spreadsheet/ database £45,Logotron Logo £35, Artisan and support disc £25. Tel. 01-341 2187 eves.

Master 512 dual 40/80T Opus DS DD. Microvitec Cub colour monitor. Acorn Teletext Receiver with ATS 3.0. View, Viewsheet, Dumpmaster, ISO Pascal fitted. 2 ROM cartridges. GEM & Nidd Valley mice. Chauffeur etc. All manuals for above including View and Viewsheet plus Dabs Guide. Shibumi Problem Solver (will run Flight Simulator ver 3.0 dBaseII and AutoRoute). Many ADFS and MS-DOS shareware programs (including printer drivers, disc cataloguer and ADFS menu) on 50 x 5.25" discs. £650 o.n.o. Tel. (0773) 823450 eves & weekends.

Brother HR5 printer in very good condition. Serial interface. Ideal for program listings, complete with Master lead. £50 o.n.o. Tel. (07048) 79902 after 6pm.

Acorn 512 board complete and as new with Dabs shareware offers? Morley Teletext Adaptor complete with manuals ROMs and separate PSU £67, Acorn Z80 second processor complete and in original box £82, Advanced Disc User Guide - Pharo £6, 27010 EPROM (1 meg) £16. Tel. 051-6475367.

512 co-processor, DOS 2.1 plus various books £90. Acorn teletext decoder £40. Two 3.5" disc drives £50 each. Tel. (0223) 358774.

Master 128, Zenith 12" monitor, 2 DS 40/80T, Morley Adaptor, Master Modem with command ROM. View, Viewsheet, Viewstore, Hyperdriver and other ROMs £425. Also vol 4 to date BEEBUG mags. Offers? Tel. (0703) 552821.

Viewsheet 16k ROM based spreadsheet, complete with fitting instructions and manual £20. Cumana model CS100 40T single sided floppy disc drive with manual £50. Both items hardly used and in original packaging. Tel. (0753) 887681.

BEEBUG magazines 6 issues vol 1, vol 2, vol 3, vol 5, vol 6, vol 7, vol 8 (to current issue), Offers? **WANTED:** Hard disc (min 20 meg) and interface suitable for Archimedes 310. Tel. (0332) 556381.

Master 128, dual 40/80T drives £450, AMX mouse, Stop Press, Extra Extra £40, EMR pro-performer + scorwriter + midi interface £125. Interword £15, A4 form designer £10. Tel. 021-350 3210.

Master Compact, complete with single disc drive, monitor stand, RGB colour monitor 2 years old. Offers? Tel. (0823) 283833.

Morley teletext adaptor with ATS 2.59 and support ROMs. User guides included £70 o.n.o. Tel. 01-644 9564.

Pinapple software PCB layout, disc ROM and manual £50. Exmon II ROM and manual, Termulator ROM and manual, Office Master Accounts disc and manual £12 each. Tel. (0206) 392168.

Shinwa CP80 unused ribbons for sale. £4 each. Tel. (0963) 63497.

POSTBAG

READER SURVEY

At the time of our reader survey, conducted earlier this year, a good many BEEBUG members took the trouble to write more fully with their comments. Some of these are reproduced below.

"Present size is convenient for pocket and book shelf."

"I would like to see more sideways RAM based utilities - it is important to make as much use as possible of this special area of memory."

"Is it possible to have some kind of check sum for each line of a program so that you can check for any mis-typings? *We have considered this on a number of occasions but not found a system which we believe has sufficient advantages.*

"BEEBUG is tending too much towards articles, reviews etc. relating mainly to the Arc". *Not so, but we do intend to help those readers who may consider upgrading to an Archimedes or A3000.*

"Don't forget BBC model B owners. Too many programs favour the Master series." - *very few programs are specific to the Master.*

"To date it has concentrated on BBC micro and Master - keep it that way."

"Some articles lack sufficient detail to operate a program once it has been typed in." *We do our best, but must try harder.*

"Reprint an outstanding *old* BEEBUG program each month for the benefit of newer members." *Back issues are still available.*

"Please try to minimise splitting of articles (and programs) in the magazine."

"More information on disc systems, more Postbag and Hints. Less listings - better on disc."

"Not enough readers' letters."

Many readers said that they wanted the magazine to stay just as it is. The aim of BEEBUG is to continue to provide the best possible support for the BBC model B, Master and Compact for as long as there is a demand for this. Most of our programs will work on all machines, and that continues to remain our aim as far as published software is concerned. We continue to welcome all contributions from readers - letters, hints & tips, articles, and programs. Indeed, most of the content of each issue consists of members' own contributions.

LOCATING A LOCAL USER GROUP

Following your recent enquiry from Emma Pithers of Coventry (Postbag Vol.8 No.5), I regret to say that we do not have a BBC group listed in that area. In fact the nearest group we have is Telford, Aylesbury or Sheffield. The predominance of BBC groups of four years ago has disappeared, and while there are still quite a few listed on our database, there isn't one in that area.

You may ask who we are. *The British Association of Computer Clubs* (BACC) is the answer. We hold a database of computer groups throughout the UK currently standing at over 1000 groups, and anyone wishing to find out their nearest local, or national user group for their machine can, for the cost of a stamp and an SAE (most important), receive three contacts.

There are groups in Coventry for Amiga, Atari, Tandy and PCs but not BBC micros. Of course, if anyone wishes to start a group, here or anywhere else, then a large A4 SAE with a 36p stamp will be returned with one of our free starter packs. The address to write to is BACC, Banc Y Rhosyn, 14 Bron Y Glyn, Bronwydd, Carmarthen, Dyfed SA33 6JB.

John Dale, BACC Database Manager

AN ESSENTIAL UPDATE

Thank you for the favourable review of our 512 Ramdisc Utilities (see Vol.8 No.5), but I would like to correct an error which appears to have crept into the review and which has already misled at least a couple of customers.

The review states that drive C: is reserved for a Winchester drive, and that a RAM disc may be set up as any of drives D: to K: only. Might I point out that with this package, any drive, be it floppy or RAM disc, can be allocated to any drive from A: to M: inclusive, with the sole exception of L:.

Robin Burton, Essential Software

Archimedes A440 entry system with well organised 20mb hard disc £1850. Original software with manuals, PC Emulator £55, System Delta V2 £40, 1st Word + £45. Tel. (0272) 736237.

Various BBC/Archimedes software titles. Games and others, including Exile, Toolkit +, G.A.C, Terramex, Pacmania etc. very cheap prices. Tel. 01-947 8831.

Master 128/512 dual 40/80T drives, Viewstore, System Delta, Mos+, Dumpmaster, Chauffeur, Graffik and mouse. AMX Stop Press, Super Art and 3rd Zicon. Windomatic, Toolkit, Elite, 3 dual cartridges. Demon modem, many discs, books all manuals, BAUs and over 50 BEEBUGS £625 o.n.o. 512 board, v2.1, Dabs Guide, Shareware vol 1, utilities £160 o.n.o. Also Shinwa CP-80 printer (Epsom comp.), 3 ribbons £100 o.n.o. Ferguson RGB TV £110 o.n.o. Tel. (045527) 4018 eves.

WANTED: Viglen console for BBC B, Galaxy software Easytalk Speech Utility ROM or Computer Concepts Text to Speech & Speech ROMs for use with Acorn Speech synthesiser, uncased DS/DD 80T drive. Manuals/ documentation essential. Tel. (0703) 227093 eves.

BBC Master, twin 40/80 drives, AMX mouse, Stop Press, Interword, Eprom programmer, Master cartridge, Voltmace joystick, books, new, £800+ will sell for £450 o.n.o. Tel. (0928) 32993 after 8pm.

Philips 12" green BM7502 monitor £45. ROM: MasterROM £15, Exmon II £10, BEEBUG C 80T £20, Toolkit £5, Sleuth £5, Slave £5, ADI £7, RamRod £10, BCPL £20, ADE Plus £20. DISC: BCPL Stand Alone Generator £10, Oxford Pascal Master 80T £20, Macrom 80T £10. TAPE: Honey Logo £2, BBCsoft Toolkit £2, PCW Software £2. Tel. (0462) 812059 eves.

WANTED: Urgently AMX mouse + Super Art for Master 128. Write to Ben Carr, Flat B4/3 floor Greenville Gardens, Shiu Fai Terrace, Stubbs Road, Hong Kong or fax Hong Kong 5-7956000. Quote price including airmail to H.K.

Spellmaster 128k ROM, boxed, manual £30. Viewsheet ROM £25. Tel. (0926) 882408 eves.

BBC Master 512, Technomatic PD800P dual 5.25" drives in plinth, Cumana CD358 dual 3.5" drives, Sanyo colour monitor, Vine micros internal ROM board, PMS Genie cartridge, AMX mouse + Superart, Viglen ROM cartridge, Overview package, DOS + Problem Solver. ROMS: Interword, Interchart, Master ROM, MOS+, Sciways. Many games and manuals. Worth over £2000, accept £700 o.n.o. Will split. Tel. 021-455 9033 after 9.30pm.

Master 128 + MOS+, Pace DS/DD 40/80T Dual Drives, Morley 'AA' ROM board/Teletext Adapter/ Eprom Programmer. Watford Quest Mouse + Paint/ Wapping Editor. CC Spellmaster. AMX Max. Care/Acorn cartridges. Joystick and software. Loads of games/serious software, mags, manuals, all boxed £550. Tel. (0489) 584649 after 6pm.

Extras for the BBC: Castle Designer £20, WE Word-Aid for Wordwise plus £12, Solidisk Sideways RAM 32k £25, WE 32k Shadow RAM card and ROM £30. Tel. 01-854 6656 extn 27 during office hours.

BEEBUG magazines, complete to date, in binders £55, plus post or collect London/St Albans. Micro User vols 1-4 complete in binders £30, A&B Computing 35 magazines and supplements in 3 binders £25, MU and AU magazine listings and other cassettes £1 each (list available) Tel. (0727) 61835.

Archimedes Acorn DTP complete, new and only £140. Tel. 01-854 6656 extn 27 during office hours.

Hewlett Packard Deskjet (laser quality) printer, extra font module includes new ink cartridges (compatible with BBC, Archimedes or PC reviewed BEEBUG 7.3) £475. Pace Linnet Modem £85, MS-DOS software on 3.5" disc all suitable for Archimedes running PC emulation. Sage Retrieve Database £60. Ashton Tate Byline Desk Top Publisher £75. Smartcom III Comms package £65. New Advanced User Guide for BBC £10, all as new and in makers boxes and include insured delivery anywhere in the UK. Tel. (0536) 20 00 94.

Epson RX80FT+ Printer with BBC cable £100 o.n.o. 16k EPROMs (27128 21v) £3.50 o.n.o. Acorn User magazines 84-89 (58 issues) Offers! Tel. (0332) 556381.

View Professional (BBC B, Master or Compact) £40, Genie Junior (Master) £10, View 2.1 (Archimedes) £20, 2-way switch (1 computer to 2 printers or vice-versa) £10, Master quad cartridge £8, ADT Master cartridge £5, manuals where applicable. Also discs, books, magazines, etc. Prices exclude postage. Tel. (0732) 862404.

Psion Organiser II, Model XP, complete with 1x16k & 1x8k DataPak, 1xWidget Software Utilities Pak (eg. prints diary direct to printer), and Comms Link (RS232). Complete with all documentation, still under guarantee (Feb'89). Worth over £200 accept £100. Tel. (0344) 861988. 9-5pm weekdays.

Master 128, DSDD 40/80, Microvitec colour monitor, Toshiba printer, plinth, Interword, BEEBUG C (complete), all BEEBUG mags, manuals, books and many games and much more £700 o.n.o. Tel. (0689) 53045.

Archimedes A310 base system, RISC OS, BEEBUG external 5.25" drive interface, TWIN, Zarch, original packaging/manuals, £600. Tel. (0785) 714913.

Master Turbo 128 + 512, Microvitec 653, twin 40/80 disc driveChameleon Palette, AMX Super Art and mouse, Overview, Tubelink basic 5, Applications, large range of software, manuals, User Guide 1&2, Advanced User Guide, various books etc. £900 o.n.o. (not splitting) (0707) 325034 eves & weekends.

BEEBUG mags to '86 Micro Users &c to '88 £7.50. Jet printer and manual £20, many tapes and software (1.2 OS) including Elite, Snowball, Graphics ROM and Lightpen £30 or £50 the lot. 512 co-processor in co-pro adaptor. M-Tec Basicand "DOS plus reference Guide" (GLENTOP) £150. Tel. (0254) 886371.

Model B issue 7 with Watford 32k RAM expansion, 2 x 400k Watford floppy drives with Watford DFS, Zenith amber monitor, HCR ROM Box with Inter series, Wordwise plus, AMX Super Art, etc. AMX mouse, Voltmace Analogue Joystick. All leads and manuals (inc. Advanced User and Advanced Disc User Guides). £300. Tel. (044282) 4543.

BBC B assorted cassettes of games/adventure + Tool Box. BEEBUG magazines1982-'89 complete sets, BBC Micro Compendium by J Rushton, Filing Systems and Databases for the BBC micro, Discovering BBC Micro Machine Code by Stephenson, The BBC Micro-James, Pascal from Basic-Brown, BBC Micro disc drives-Bagnall, BBC Micro Assembly Language-Smith, 100 Programmes for the BBC Micro Computer, Century computer programming course, all as new. Offers? Tel. (0892) 30071.

Invaluable Utilities for the BBC micro £2, Basic Programming on the BBC Microcomputer £2, Getting More from your BBC and Electron Computers £2, The BBC Micro Revealed £2.50, The BBC Microcomputer Disc Companion £3, Using BBC Basic £3, The Advanced Disc User Guide for the BBC Micro £5. Tel. (0442) 62271.

Interbase 1.0A £30 o.n.o. Interbase 2.0A £35 o.n.o. Hi-Intersheet disc £4 o.n.o. (needs Intersheet installed), HCR ROM board Offers? Tel. (0707) 45953 eves.

Watford MkII 1770 double density DFS £30. Tel. (0908) 611504.

BBC Master 128, Watford Twin DS 40/80 DD Fidelity CM14 MED RES colour monitor, ROMs, Spellmaster, Dumpmaster, Command. Software Masterfile ADFS, Printwise, 1 year BEEBUG discs, 2 years BEEBUG mag Acorn Reference manuals 1&2 for Master a few games. Care cartridge. £600. Tel. 01-886 4186 MAG90832. MBX011154871.

Road Runner, Escape from Moonbase Alpha, Galaxy Wars, 3D Space Ranger, Thunderstruck 2, all tapes £1.50 each Cumana 40T SS SD with PSU £35, Martech Tarzen £3.50 disc Repton Infinity (used about 3 times) (discs & books) £8.50 Doctor Who and the mines of Terror (ROM + disc + books) £8.50. Tel. (0734) 771887 after 6pm.

WANTED: Instant Mini Office II on ROM 4 units wanted. Tel. (0929) 424175.

Master 512k, mouse, Microvitec colour monitor, Watford dual 5.25" disc drives in Master bridge, MOS+, Hyperdriver, Overview (2), Dumpmaster ROMs + cartridges, GEM, Shareware, Sidewriter, Masterfile II, Acornsoft Hits 1, Gemini, Micro Aid, Dabs discs, miscellaneous programs, various books, all manuals, BEEBUG back issues, worth £1300, sell £750 o.n.o. Tel. (0536) 83898.

Stop Press (M128), Intersheet, £20 each. Logotron Logo £25. Tel. 01-969 7294 eves.

Acorn DTP as new £120, Clares Render Bender £45, Minerva System Delta Plus (RISC OS version) £45. Tel. (0703) 634288 extn 2490 after 6pm.

BBC B OS 1.2, Opus double sided 40/80 disc drive cassette player, Mini Office II ROM, View ROM, (all with instruction manuals), joysticks, many games inc. Snapper, Chess, Elite. BEEBUG magazines (with index) from vol 1 no. 7 to date £225. Tel. 01-942 8063.

512k Board + DOS 2.1 + manual, mouse and GEM software £100. 65C102 Turbo co-processor board disc and manual £80. Tel. (0454) 778061 after 6pm.

Fun School 2, complete, 3 cassettes + handbooks, unused & unwanted gift £5. Tel. (0920) 465406.

Interbase (unused & unregistered) £45, Interword (Arc disc unregistered) £25, System Delta with card index + reference manual £45. Tel. 01-515 4686.

Master 128 in Viglen console, remote keyboard, 2 x 40/80T drives £480. BBC issue 7 £300. Aries 20 and B32 boards £85. 2 x 40/80 Technomatic drives in monitor plinth £150. Interword £30. Spellcheck £35. Microvitec Cub med. res. monitor £150. Philips 8833 med. res. monitor £185. Morley Electronics V2 EPROM blower £25. Ground Control UVIPAC EPROM eraser £20.

Mega 3 ROM from Computer Concepts for Master Compact £25, Printwise £5, Mini Office II £5, both 3.5" disc. Tel. (0428) 793040.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£16.90	1 year (10 issues) UK, BFPO, Ch.I	£25.00
£24.00	Rest of Europe & Eire	£36.00
£29.00	Middle East	£43.00
£31.00	Americas & Africa	£46.00
£34.00	Elsewhere	£51.00

BACK ISSUE PRICES (per Issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

Printed by Newnorth-Burt Ltd (0234) 41111 ISSN - 0263 - 7561

Magazine Disc/Cassette

DECEMBER 1989 DISC/CASSETTE CONTENTS

★ **DRAINSTORM** - the first of two bonus items for this issue is a challenging yet novel platform game. Excellent graphics and animation provide a highly professional presentation.

★ **STELLAR DISPLAY** - the second of our two bonus items allows all the major constellations to be selected and displayed in the night sky as seen from a specified observer position on earth.

LAUNCH A SPACECRAFT - explore the world of satellites and spacecraft with this interesting program.

EDIKIT (Part 1) - complete working program which incorporates the first of several utilities for programmers, a *find text* function.

DISC FILE IDENTIFIER (Part 2) - the complete program combining part one from last month, and the ADFS and WOTAMI additions from part two.

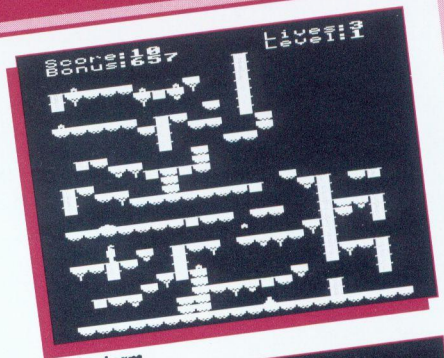
A FIND UTILITY - a short utility which allows any file on disc to be searched for a specified text string.

SOLIDS OF REVOLUTION - use this enterprising program to investigate the world of 3D graphics and solids of revolution. Sample data files are included.

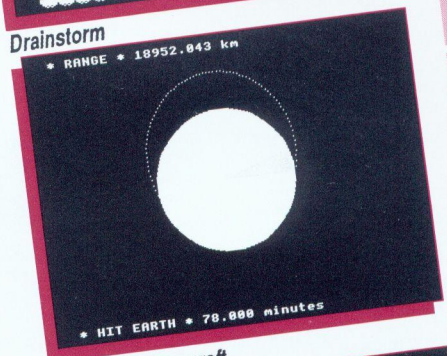
POSTSCRIPT VECTOR GRAPHICS - an alternative approach to producing BBC micro graphics in PostScript format, complete with test and demo programs.

AMATEUR RESEARCH (Part 3) - this month's program explores the two-body and many-body problems of gravitational attraction.

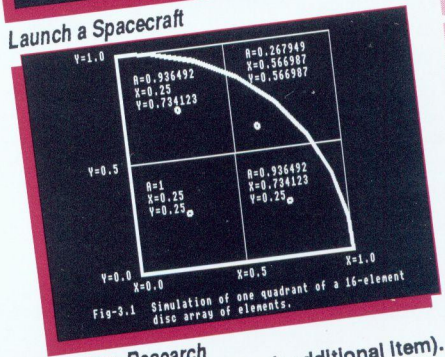
MAGSCAN - bibliography for this issue (Vol.8 No.7).



Drainstorm



Launch a Spacecraft



Amateur Research

All this for £3. 50 (cassette), £4.75 (5" & 3.5" disc) + 60p p&p (30p for each additional item).
Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY			OVERSEAS		
	5" Disc	3.5" Disc	Cassette	5" Disc	3.5" Disc	Cassette
6 months (5 issues)	£25.50	£25.50	£17.00	£30.00	£30.00	£20.00
12 months (10 issues)	£50.00	£50.00	£33.00	£56.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

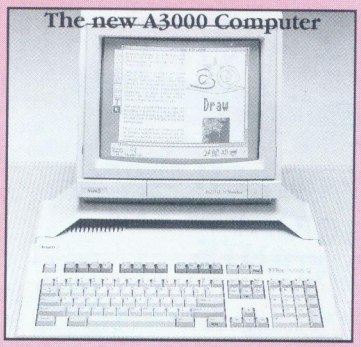
Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS.

BEEBUG - THE ARCHIMEDES SPECIALIST

The NEW BBC Micro - The A3000

Ideal for home, education and business use, this powerful computer includes the multi-tasking Operating System, RISCOS. BBC Basic V and a BBC B emulator are supplied as standard. An optional PC emulator disc allows over 90% of DOS software to be used including Lotus, dBase III, MS Word and Wordperfect.



FREE ON-SITE MAINTENANCE

All new Archimedes and A3000 computers purchased through Beebug include one years On-Site Maintenance.
Beebug is the only Acorn dealer offering this service.

0% Finance

Under this scheme you may purchase an Archimedes A3000, 300 or 400 Series computer (base, mono or colour) with an initial deposit and 9 payments at monthly intervals. There is no charge to you for this service.

Some examples of the repayments are:

Machine	Deposit	9 Payments
A3000	71.35	75.00
A3000 Colour	99.70	96.00
310	106.85	103.00
310 Colour	126.20	125.00
310M	112.85	110.00
310M Colour	132.20	132.00
410/1	145.85	137.00
410/1 Colour	165.20	159.00
420/1	198.85	195.00
420/1 Colour	218.20	217.00
440/1	290.85	287.00
440/1 Colour	310.20	309.00

If you wish to take advantage of this scheme please telephone us and we will send you further details. Finance over 12/24/36 months is also available.

THESE PRICES INCLUDE VAT

SEE BELOW FOR SPECIAL OFFERS

ACORN A3000 SERIES

0255G	A3000 Entry System	649.00
0256G	A3000 Colour System	838.00

ARCHIMEDES 300 SERIES

0193G	310 Entry System	899.00
0195G	310 Colour System	1088.00

ARCHIMEDES 400 SERIES

0260G	410/1 Entry System	1199.00
0261G	410/1 Colour system	1388.00
0262G	420/1 Entry System	1699.00
0264G	420/1 Colour System	1888.00
0275G	440/1 Entry System	2499.00
0276G	440/1 Colour System	2688.00

These prices exclude VAT

SPECIAL OFFER ON COLOUR SYSTEMS

We are now able to offer a choice of either Acorn colour monitor or Phillips colour monitor.

SPECIAL DISCOUNTS FOR TEACHERS

We are currently able to offer significant discounts to teachers and Acorn Shareholders on computers. Please phone for further information.

RISC USER & BEEBUG MEMBERS BUYING AN ARCHIMEDES SYSTEM WILL RECEIVE FREE:

A3000 -On Site Maintenance, and items to the value of £46 (entry system) or £56 (colour system). Members using the 0% finance offer receive On-Site Maintenance only.

310 - PC Emulator, Printer Lead, Ten 3.5" discs & Lockable disc box.

410/1 - On Site Maintenance, and items to the value of £105 (Entry system) or £120 (Colour system). Members using the 0% finance offer receive free On-Site Maintenance only.

420/1- On Site Maintenance, and items to the value of £166 (Entry system) or £177 (Colour system). Members using the 0% finance offer receive On-Site Maintenance and 10 x 3.5" discs.

440/1 - On Site Maintenance and items to the value of £259 (Entry system) or £269 (Colour system). Members using 0% finance receive free On Site Maintenance, 10 discs in lockable Box & Interdictor

Please phone or write to receive a copy of our new FREE 40 page retail catalogue.

117 Hatfield Road, St Albans Herts AL1 4JS Tel: (0727) 40303 Fax: (0727) 60263